TESIS – KI142502

PENEMUAN WEB SERVICE BEDASARKAN KESAMAAN STRUKTUR, SEMANTIK, DAN PERILAKU

Andreyan Rizky Baskara
5115201029

DOSEN PEMBIMBING
Prof. Drs.Ec. Ir. Riyanarto Sarno M.Sc., Ph.D

PROGRAM MAGISTER
BIDANG KEAHLIAN REKAYASA PERANGKAT LUNAK
JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2017

THESIS – KI142502

WEB SERVICE DISCOVERY BASED ON STRUCTURE, SEMANTIC AND BEHAVIOR SIMILARITY

Andreyan Rizky Baskara
5115201029

SUPERVISOR
Prof. Drs.Ec. Ir. Riyanarto Sarno M.Sc., Ph.D

MAGISTER PROGRAM
SOFTWARE ENGINEERING
DEPARTMENT OF INFORMATICS ENGINEERING
FACULTY OF INFORMATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2017

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar

Magister Komputer (M.Kom.)

Di

Institut Teknologi Sepuluh Nopember Surabaya

Oleh:

Andreyan Rizky Baskara

NRP. 5115201029

Dengan judul :

PENEMUAN WEB SERVICE BERDASARKAN KESAMAAN STRUKTUR, SEMANTIK DAN PERILAKU

Tanggal Ujian : 11 – 7 – 2017

Periode Wisuda : 2016 Genap

Disetujui oleh:

Prof.Ir.Drs.Ec. Riyanarto Sarno, M.Sc, Ph.D
NIP. 195908031986011001
(Pembimbing 1)

Daniel Oranova Siahaan, S.Kom, M.Sc, PD.Eng
NIP. 197411232006041001
(Penguji 1)

Sarwosri, S.Kom, M.T
NIP. 197608092001122001
(Penguji 2)

Adhatus Sholichah, S.Kom, M.Sc
NIP. 198508262015042002
(Penguji 3)

Dekan Fakultas Teknologi Informasi,

Dr. Agus Zainal Arifin, S.Kom., M.Kom.
NIP. 197208091995121001

PENEMUAN WEB SERVICE BERDASARKAN KESAMAAN STRUKTUR, SEMANTIK DAN PERILAKU

Nama	: Andreyan Rizky Baskara
NRP	: 5115201029
Pembimbing 1	: Prof. Drs.Ec. Ir. Riyanarto Sarno M.Sc., Ph.D.

ABSTRAK

Web Service mempunyai peran penting pada lingkungan komputing bisnis saat ini untuk mengembangkan aplikasi yang terdistribusi di berbagai jaringan. Banyak pelaku bisnis yang sudah menerapkan proses bisnisnya menggunakan Web Service. Pelaku bisnis tersebut dapat mengganti Web Service yang digunakannya dikarenakan perubahan rekan bisnis atau berhenti berjalan (offline). Seiring waktu, Jumlah web service yang sudah dipublikasikan di web semakin meningkat. Dengan banyaknya web service yang beredar di web, penemuan Web Service yang optimal diperlukan untuk memenuhi kebutuhan Web Service sesuai keinginan user.

Web Service mempunyai dua tipe, yaitu atomik dan komposit. Untuk penemuan Web Service atomik, pengukuran kesamaan semantik dan struktur diperlukan. Sedangkan untuk penemuan Web Service komposit, pengukuran kesamaan semantik dan perilaku diperlukan. Untuk penemuan Web Service atomik, dokumen Web Service Definition Language (WSDL) digunakan sebagai masukan dan untuk penemuan Web Service komposit, Business Process Execution Language (BPEL) digunakan sebagai masukan. Penelitian sebelumnya menggunakan Latent Dirichlet Allocation untuk mengukur kesamaan semantik berdasarkan topik. Namun LDA memiliki kelemahan ketika digunakan untuk mengekstraksi topik pada Web Service.

Beberapa Web Service yang dideskripsikan dengan WSDL atau BPEL, apabila diekstraksi informasinya maka akan membentuk dokumen pendek. Ketika LDA digunakan untuk mencari kesamaan semantic pada WSDL dan BPEL, metode ini tidak dapat berjalan dengan baik dikarenakan keberagaman kata pada dokumen pendek tidak terlalu banyak sehingga dapat mempengaruhi hasil penggalian topik. Sehingga diperlukan metode varian LDA yang mampu menggali topik pada dokumen pendek yaitu Biterm Topic Model (BTM). Penelitian ini mengusulkan untuk menggunakan WDAG untuk memodelkan struktur dan perilaku Web Service dan menggunakan WDAG Similarity untuk mengukur kesamaan struktur atau perilakunya. BTM digunakan untuk mengukur kesamaan topik antar label *node* pada WDAG.

Analisis pengujian akan dilakukan dengan menggunakan metriks *Precision* dan *Recall*. *Precision* dan *Recall* adalah dua perhitungan yang banyak digunakan untuk mengukur kinerja dari sistem atau metode yang digunakan pada bidang sistem temu kembali (Information Retrieval). Pendekatan yang diusulkan diharapkan dapat mengurai *false positive* dan *false negative* saat melakukan penemuan Web Service sehingga dapat meningkatkan nilai *Precision* dan *Recall*.

Kata Kunci: Perilaku, Penemuan Web Service, Semantik, Struktur, Topic Model.

[halaman sengaja dikosongkan]

Web Service Discovery based on Structure, Semantic and Behavior Similarity

Student's Name	: Andreyan Rizky Baskara
NRP	: 5115201029
1st Supervisor	: Prof. Drs.Ec. Ir. Riyanarto Sarno M.Sc., Ph.D

ABSTRACT

Web Service has an important role on business computing in order to develop distributed application in networks. Many business actors have implemented their business processes using web service. Those business actors can change their web service depending on their conditions like when they change their business partner or their Web Service goes offline. Now, A lot of web services have been published on the web. With this condition, A need of optimal Web Service discovery is increasing in order to find a Web Service that satisfies a business actor.

Web Service have two types, which are atomic Web Service and composite Web Service. To discover atomic web services, semantic and structure similarity are needed. To discover composite web services, semantic and behavior similarity are needed. Web Service Definition Language (WSDL) is used as an input on atomic Web Service discovery processes. Business Process Definition Language (BPEL) is used as an input on composite Web Service discovery processes. Previous research used Latent Dirichlet Allocation (LDA) to measure semantic similarity of web services based on topic. But LDA has weakness.

Web services are descripted using WSDL or BPEL. When information inside WSDL or BPEL extracted, then some of it will create a short document. LDA performance is not going well on short document because of word co-occurrence sparsity. The word co-occurrence sparsity may affect the performance of LDA in a bad way. As of another variant of LDA that can mine topic on short documents is needed such as Biterm Topic Model (BTM). This research proposed on using Weighted Directed Acyclic Graph (WDAG) to models structure and behaviour of a Web Service and using WDAG Similarity to calculate the similarity of structure and behaviour between web services. Biterm Topic Model is used to mine the latent topic on node label of WDAG and calculate its semantic similarity.

The test results are analysed by using Precision and Recall. Precision and Recall are two measurement that mostly used to assess performance of system or method which is used in Information Retrieval. The proposed approach in this research is expected to reduce false positive and false negative in order to increase the Precision and Recall value.

Keywords: Behavior, Semantic, Structure, Topic Model, Web Service Discovery

[halaman sengaja dikosongkan]

KATA PENGANTAR

Segala puji bagi Allah SWT, yang telah melimpahkan rahmat dan hidayah-Nya sehingga penulis bisa menyelesaikan Tesis yang berjudul "Penemuan Web Service Berdasarkan Kesamaan Struktur, Semantik, dan Perilaku" sesuai dengan target waktu yang diharapkan.

Pengerjaan Tesis ini merupakan suatu kesempatan yang sangat berharga bagi penulis untuk belajar memperdalam ilmu pengetahuan. Terselesaikannya buku Tesis ini, tidak terlepas dari bantuan dan dukungan banyak pihak. Oleh karena itu, penulis ingin menyampaikan rasa terima kasih yang sebesar-besarnya kepada:

1. Allah SWT atas limpahan rahmat-Nya sehingga penulis dapat menyelesaikan Tesis ini dengan baik.
2. Bapak H. Basuki dan Ibu Hj. Rahmi Widyanti, selaku orang tua penulis yang selalu mendoakan agar selalu diberikan kelancaran dan kemudahan dalam menyelesaikan Tesis ini. Serta menjadi motivasi terbesar untuk mendapatkan hasil yang terbaik.
3. Bapak Prof. Drs.Ec. Ir. Riyanarto Sarno M.Sc., Ph.D, selaku dosen pembimbing yang telah memberikan kepercayaan, motivasi, bimbingan, nasehat, perhatian serta semua bantuan yang telah diberikan kepada penulis dalam menyelesaikan Tesis ini.
4. Bapak Daniel Oranova Siahaan, S.Kom, M.Sc, PD.Eng, Ibu Sarwosri, S.Kom, M.T, dan Ibu Adhatus Solichah, S.Kom, M.Sc selaku dosen penguji yang telah memberikan bimbingan, saran, arahan, dan koreksi dalam pengerjaan Tesis ini.
5. Bapak Waskitho Wibisono, S.Kom., M.Eng., PhD selaku ketua program pascasarjana Teknik Informatika ITS, Bapak Royyana Muslim I, S.Kom., M.Kom., Ph.D., selaku dosen wali penulis dan segenap dosen Teknik Informatika yang telah memberikan ilmunya.

6. Mbak Lina, Mas Kunto dan segenap staf Tata Usaha yang telah memberikan segala bantuan dan kemudahan kepada penulis selama menjalani kuliah di Teknik Informatika ITS.
7. Kakak penulis, Selviana Rizky Pramitha, serta seluruh keluarga besar yang selalu memberi semangat, doa, dukungan dan hiburan kepada penulis.
8. Rekan seperjuangan tesis 2017 yang selalu menjadi teman diskusi yang baik, serta selalu mendo'akan dan mendukung penulis. Insya Allah rekan-rekan semua diberikan kemudahan dalam penyelesaian tesis.
9. Rekan-rekan angkatan 2015 Pasca Sarjana Teknik Informatika ITS yang telah menemani dan memberikan bantuan serta motivasi untuk segera menyelesaikan Tesis ini.
10. Juga tidak lupa kepada semua pihak yang belum sempat disebutkan satu per satu disini yang telah membantu terselesaikannya Tesis ini.

Sebagai manusia biasa, penulis menyadari bahwa Tesis ini masih jauh dari kesempurnaan dan memiliki banyak kekurangan. Sehingga dengan segala kerendahan hati, penulis mengharapkan saran dan kritik yang membangun dari pembaca.

Surabaya, 19 Juli 2017

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Web Service mempunyai peran penting pada lingkungan komputing bisnis saat ini untuk mengembangkan aplikasi yang terdistribusi di berbagai jaringan. Web Service adalah sebuah komponen perangkat lunak yang dapat dikembangkan, dideskripsikan, dipublikasikan, ditemukan, digunakan dan digunakan kembali untuk mencapai sebuah fungsionalitas yang spesifik. Pelaku bisnis dapat memanfaatkan aset mereka dengan mempublikasikan Web Service atau Application Programming Interfaces (API) nya, sehingga memungkinkan perusahaan tersebut untuk meningkatkan skala penggunaan fungsionalitasnya ke tingkat web dan menggunakan berbagai macam bisnis model yang berbeda-beda. Pelaku bisnis dapat mengganti Web Service yang mereka gunakan berdasarkan kondisi mereka, misalkan perubahan rekan bisnis atau Web Service yang sedang digunakan berhenti berjalan (*offline*).

Web Service didefinisikan sebagai sebuah sistem perangkat lunak yang didesain untuk mendukung interoperabilitas interaksi mesin dengan mesin di jaringan. Jumlah Web Service yang sudah dipublikasikan di web semakin meningkat seiring waktu. Sebagai contoh, Website Programmableweb telah mempublikasikan sekitar 16.780 web service pada Januari, 2017 (Wang et al. 2015). Dengan banyaknya Web Service yang beredar di web, pencarian Web Service yang optimal diperlukan untuk memenuhi kebutuhan Web Service sesuai keinginan user (Maheswari 2014). Sehingga diperlukan cara untuk mengurangi false positive dan false negative dalam penemuan Web Service.

Berbagai macam metode diterapkan untuk kasus penemuan Web Service, salah satunya adalah penemuan web service berdasarkan kata kunci atau kesamaan sintaksis (Qin et al. 2010). Namun, pencarian bedasarkan kemiripan kata kunci kurang efektif dikarenakan banyak deskripsi tekstual Web Service yang tidak lengkap dan juga penggunaan kata sinonim atau variasi dari kata kunci yang

ditetapkan. Kata yang digunakan pada fitur Web Service mungkin berbeda secara sintaksis namun memiliki kesamaan makna atau semantik.

Pendekatan untuk penemuan Web Service dengan mengukur derajat kesamaan antar service dapat dilakukan dengan tiga cara, yaitu pengukuran kesamaan semantik, struktur, dan *behavior* (Bravo 2014). Pendekatan dengan pengukuran kesamaan semantik adalah dengan mengukur derajat kesamaan kontekstual antar Web Service. Hal ini dapat dilakukan dengan berbagai macam cara, salah satunya dengan metode Information Retrieval. Metode IR yang sering digunakan adalah Vector Space Model (VSM) (Halevy et al. 2004; Platzer and Dustdar 2005) dan Latent Semantic Analysis (LSA) (Hou et al. 2010; Wu, Potdar, and Chang 2008). Meskipun VSM dan LSA simple dan mudah diimplemetasikan, Metode – metode ini memiliki kelemahan.

VSM menggunakan Term Frequency-Inverse Document Frequency (TF-IDF) untuk pembobotannya. TF-IDF menggunakan jumlah kata yang muncul dalam dokumen untuk selanjutnya dibandingkan dengan dokumen lain. Sehingga, apabila ada kata yang berbeda tapi memiliki maksud yang sama, misalkan, kata apel dan jeruk, sama-sama buah tapi beda penulisan kata, maka akan menurunkan nilai kesamaan.

LSA menggunakan Singular Value Decomposition (SVD) untuk mencari korelasi antar kata. Hal ini ditujukan untuk menutupi kelemahan TF-IDF yang tidak dapat membedakan kata yang beda tapi memiliki maksud yang sama. Namun, kompleksitas LSA akan semakin bertambah tergantung banyaknya jumlah kata. Dan hasil SVD dari LSA tidak dapat dinterpretasikan nilainya.

Peneliti pada bidang Information Retrieval kemudian mengembangkan suatu pendekatan yang dinamakan pemodelan topik. Salah satu metode dalam pemodelan topik adalah Latent Dirichlet Allocation (LDA) (Blei, Ng, and Jordan 2003). Ide dari LDA adalah, sebuah dokumen merupakan campuran dari berbagai macam topik. Dan satu topik itu merupakan campuran probabilitas dari banyak kata. LDA dibuat untuk melengkapi kekurangan dari metode IR sebelumnya dimana VSM, probabilistic IR, dan LSI tidak dapat membedakan kata sinonim dan polisemi. Metode LDA telah diimpelementasikan oleh beberapa peniliti dalam bidang penemuan Web Service (Aznag, Quafafou, and Jarir 2013; Lei and Yu 2016;

Li et al. 2013; Zhang et al. 2016). Namun LDA sendiri memiliki keterbatasan dalam mengolah teks yang pendek (Yan et al. 2013).

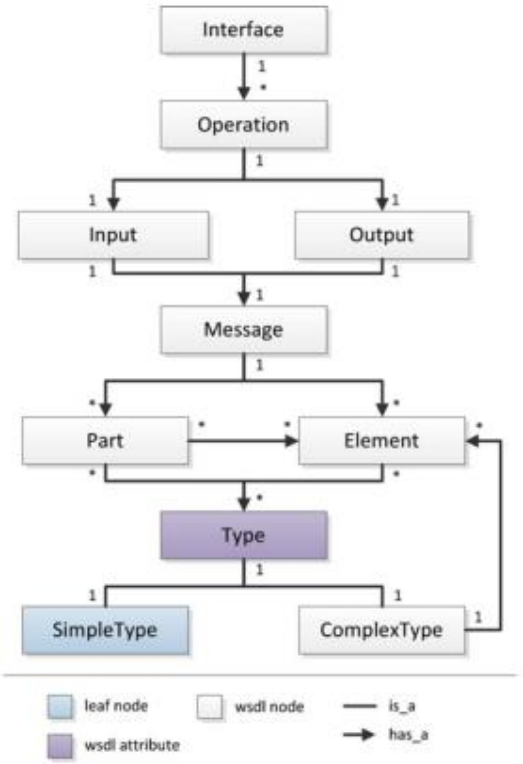
Dalam hasil observasi studi kasus, Web service didefinisikan dengan menggunakan WSDL (Web Service Definition Language). Apabila diekstraksi informasi dari WSDL mengenai nama servis, input, output, dan operation, maka hasil ekstraksi tersebut akan membentuk sebuah dokumen pendek. Ketika LDA diterapkan untuk melakukan ekstraksi topik pada dokumen tersebut, maka akan berdampak pada performanya, dikarenakan sedikitnya keberagaman kata dalam dokumen tersebut. Beberapa penelitian telah dilakukan untuk mengatasi kelemahan LDA pada dokumen pendek, salah satunya adalah metode Bi-Term Topic Model (Yan et al. 2013).

Pendekatan penemuan Web Service yang kedua yaitu dengan mengukur kesamaan struktur. Pengukuran dilakukan dengan memanfaatkan struktur XML dari WSDL (Zinnikus, Rupp, and Fischer 2010). Gambar 1.1 menunjukkan ilustrasi dari struktur WSDL. Setiap WSDL pasti memiliki Operation, Setiap Operation memiliki Input dan Output, setiap Input dan Output memiliki Message, setiap Message memiliki Part dan Element (Stavropoulos et al. 2016). Berdasarkan Gambar 1.1, struktur WSDL diekstraksi menjadi Tree dan kemudian dihitung derajat kesamaannya.

Web Service mempunyai dua tipe, yaitu atomic dan komposit. Web service atomic adalah web service yang diimplementasikan hanya memanggil operasi yang ada didalamnya. Sedangkan web service komposit adalah web service yang diimplementasikan dengan memanggil service lain yang bukan bagian dirinya. Untuk melakukan penemuan web service atomic, dengan mengukur kesamaan semantic dan structural dibilang sudah cukup, sedangkan untuk penemuan web service komposit harus diperlukan pengukuran kesamaan pada aspek lain, yaitu kesamaan *behavior*.

Pendekatan penemuan Web Service yang ketiga, yaitu dengan mengukur perilaku dari Web Service. Pengukuran perilaku web service yang dimaksud adalah dengan melakukan proses pencocokan pada model *behavior* (Grigori et al. 2010). Model *behavior* dari Web Service dapat dilihat pada BPEL (Bussiness Prosess

Execution Language). BPEL diekstraksi menjadi graph dan kemudian diukur kesamaan graph antar Web Service.



Gambar 1.1 Struktur XML dalam WSDL

Oleh karena itu, Pada penelitian ini diusulkan pendekatan baru untuk penemuan Web Service dengan melakukan pengukuran kesamaan semantik, struktural, dan *behavior*al. Untuk kesamaan semantik, pendekatan ini menggunakan Bi-Term Topic Model (BTM) untuk mengekstraksi topik pada Web Service. Metode BTM ini dikembangkan untuk menyelesaikan masalah pengekstraksian topik pada dokumen pendek pada studi Information Retrieval. Untuk kesamaan struktur dan *behavior*, pendekatan ini menggunakan Weighted Directed Acyclic Graph Similarity. Pendekatan yang diusulkan memiliki 3 tahapan. Tahap pertama adalah pembuatan Weighted Directed Acyclic Graph (WDAG), yaitu informasi penting tentang web service pada WSDL dan BPEL diekstraksi dan kemudian struktur WSDL dan *behavior* BPEL dibangun menjadi WDAG. Kemudian WDAG yang telah dibangun diserialisasikan ke dalam bentuk RuleML. Tahap kedua, pengukuran similaritas, yaitu mengukur kesamaan semantik antar

label *node* pada WDAG dan kemudian mengukur kesamaan struktur dan kesamaan perilaku antara kueri pengguna dan Web Service direpositori menggunakan WDAG Similarity. Tahap ketiga, perangkingan nilai kesamaan secara keselurahan dan menampilkan web service mana saja yang relevan.

1.2 Perumusan Masalah

Berdasarkan latar belakang tersebut, perumusan masalah dalam penelitian ini adalah sebagai berikut.

1. Bagaimana mengembangkan metode penemuan web service berdasarkan kesamaan semantik dan struktur menggunakan Biterm Topic Model dan Weighted Directed Acyclic Diagram Similarity?
2. Bagaimana mengembangkan metode penemuan web service berdasarkan kesamaan semantik dan *behavior* menggunakan Biterm Topic Model dan Weighted Directed Acyclic Diagram Similarity?
3. Bagaimana mengurangi *false positive* dan *false negative* pada penemuan web service untuk meningkatkan *Precision* dan *Recall*?

1.3 Batasan Masalah

Untuk menghindari meluasnya permasalahan yang akan diselesaikan, maka dalam penelitian memiliki beberapa batasan, yaitu:

1. *File* yang digunakan adalah WSDL dan BPEL
2. Untuk mengukur kesamaan *behavior* pada BPEL, diasumsikan *input* dan *output* pada BPEL adalah sama.
3. Bobot pada WDAG dibagi sama rata berdasarkan jumlah *node* anaknya dengan total sama dengan 1.
4. Jumlah topik yang akan digali menggunakan metode BTM sudah ditentukan sebelumnya.
5. Nilai *threshold* untuk pengembalian Web Service yang mirip berdasarkan nilai similaritas WDAG sudah ditentukan sebelumnya.

1.4 Tujuan Penelitian

Tujuan pada penelitian ini adalah untuk mengurangi *false positive* dan *false negative* pada penemuan Web Service sehingga dapat meningkatkan *Precision* dan *Recall*.

1.5 Manfaat Penelitian

Pendekatan yang diusulkan diharapkan dapat membantu pengguna dalam menemukan web service yang mirip secara tepat dengan mengukur kesamaan struktur, semantik dan perilaku.

1.6 Kontribusi Penelitian

Kontribusi dari penelitian ini adalah mengusulkan pendekatan penemuan web service dengan mengukur kesamaan semantik, struktur, dan *behavior*. Pada pendekatan ini, web service yang berupa WSDL akan dibangun strukturnya menjadi Weighted Directed Acyclic Graph (WDAG) dan web service yang berupa BPEL akan dibangun *behavior*-nya menjadi WDAG juga. Setelah itu dilakukan pengekstraksian topik pada masing-masing *node* label yang ada di WDAG menggunakan metode Biterm Topic Model. Kemudian dihitung kesamaan topik antar keduanya dengan menggunakan Jenson-Shannon Divergence sehingga dapat diketahui kesamaan semantik *node* labelnya berdasarkan topik. Kemudian dihitung kesamaan struktur atau *behavior* menggunakan WDAG Similarity.

BAB 2
KAJIAN PUSTAKA

Pada bab ini akan diuraikan mengenai konsep dasar tentang teori yang akan digunakan dalam penelitian ini. Pemaparan tersebut meliputi penjelasan tentang Penemuan Web Service, Web Service, Web Service Definition Language (WSDL), Business Process Execution Language (BPEL), Bi-Term Topic Model, Weighted Directed Acylic Graph Similarity, dan pengukuran kesamaan Jensen-Shanon Divergence.

2.1 Penemuan Web Service

Penemuan Web Service merupakan sebuah aktivitas untuk mendapatkan Web Service yang cocok dan memenuhi kebutuhan fungsional maupun non fungsional pengguna dari sekumpulan web service (Paul, Roy, and Hussain 2016). Untuk dapat menemukan Web Service yang sesuai, kesamaan dalam beberapa aspek perlu diukur. Beberapa aspek yang diukur kesamaannya pada Web Service untuk penemuan Web Service adalah semantik, struktur, dan perilaku (*behavior*).

Banyak cara diterapkan untuk melakukan penemuan Web Service. Tabel 2.1 menjelaskan ulasan penelitian terdahulu oleh beberapa peniliti yang telah melakukan proses penemuan Web Service beserta ulasan penelitian yang dilakukan pada tesis ini.

Tabel 2.1 Ringkasan penelitian pada studi penemuan Web Service

Peneliti	Tahun	Aspek	Metode	Ulasan
Halevy et al	2004	Sintaksis	Vector Space Model	VSM menggunakan Term Frequency-Inverse Document Frequency (TF-IDF) untuk pembobotannya. TF-IDF merupakan jumlah kemunculan kata pada dokumen. VSM tidak

				menggali semantik pada dokumen.
Wu et al	2008	Semantik	LSA	LSA menggunakan Singular Value Decomposition (SVD) untuk mentransformasi TF-IDF ke bidang latent. Hasil keluaran dari SVD tidak dapat diinterpretasikan nilainya dan kompleksitas LSA semakin bertambah sesuai banyak kata.
Zinnikus et al	2010	Sintaksis dan Struktur	Tree Matching	Tidak menggunakan semantic matching dan menggunakan string matching untuk pengukuran kesamaan label pada tree sehingga apabila ada kata yang berbeda tapi memiliki makna sama maka akan menurunkan nilai kesamaan.
Grigori et al	2010	Sintaksis dan Struktur	Graph Matching	Tidak menggunakan semantic matching dan menggunakan string matching untuk pengukuran kesamaan label pada graph sehingga apabila ada kata yang berbeda tapi memiliki makna sama maka akan

				menurunkan nilai kesamaan.
Lei et al	2016	Semantik	LDA	Apabila LDA diterapkan untuk melakukan ekstraksi topik pada dokumen pendek, maka akan berdampak pada performanya, dikarenakan sedikitnya keberagaman kata dalam dokumen tersebut.
Tesis ini	2017	Semantik, Struktur, dan *Behavior*	BTM dan WDAG Similarity	BTM digunakan untuk mengukur kesamaan semantik pada kata berdasarkan kesamaan topik, sehingga diharapkan dapat menemukan kata sinonim. Apabila menggunakan pengukuran semantik saja, maka informasi mengenai struktur dan *behavior* dari WSDL atau BPEL diabaikan sehingga digunakan WDAG untuk memodelkan struktur dan *behavior* dan diukur kesamaannya menggunakan WDAG Similarity.

2.1.1 Kesamaan Semantik

Menurut Kamus Besar Bahasa Indonesia, Semantik dapat diartikan sebagai makna. Sesuatu dilihat secara semantik berarti bahwa sesuatu tersebut dilihat dengan menggali maknanya. Makna sebuah kalimat tergantung dari makna dari masing-masing kata dalam kalimat tersebut. Hubungan antara makna setiap kata dalam kalimat akan memperkaya arti sebuah kalimat. Prinsip semantik dapat diterapkan dalam aktifitas penemuan web service.

Penerapan prinsip semantik dalam aktifitas penemuan Web Service adalah dengan menggali makna untuk setiap kata pada operasi, input, dan output pada web service. Web Service dapat didefinisikan menjadi sebuah dokumen Web Service Definition Language (WSDL). Dari dokumen WSDL tersebut, informasi penting pada web service dapat diekstraksi seperti misalkan nama operasi, nama input, nama output dan tipe message.

Contoh dokumen WSDL yang mendeskripsikan sebuah Web Service yaitu web service LocationMap dan PositionMap ditunjukkan pada Gambar 2.1. Apabila dilakukan perhitungan similaritas menggunakan perhitungan sintaksis seperti VSM, maka kata Location dan Position dikatakan tidak sama, namun secara bahasa kedua kata tersebut memiliki arti yang mirip. Sehingga hal ini dapat menyebabkan terjadi nya *false positive* yaitu Web Service yang seharusnya relevan dikatakan tidak relevan.

Dokumen WSDL dijadikan sebagai masukan untuk menghitung kesamaan semantik antar Web Service. Hasil ekstraksi dari dokumen WSDL yang ditunjukan pada Gambar 2.1 dapat dilihat pada Gambar 2.2. Gambar 2.2 menunjukkan kata-kata tekstual tentang nama *service*, nama *input* dan *output*, nama *message*, dan tipe *message* yang sudah diekstraksi dari WSDL LocationMapInterface dan PositionMapInterface. Selanjutnya untuk mengukur kesamaan semantik, setiap kata yang sudah diekstraksi akan digali makna atau artinya. Pada penelitian ini, aspek semantik pada WSDL yang dimaksud adalah sebaran topik yang dihasilkan dari metode BTM. Sehingga kesamaan semantik yang diukur adalah kesamaan sebaran topik pada dokumen WSDL antar web service.

```
<types><schema ...>
  <simpleType name="LocationType" modelReference=
      "http://...#Location">
    <restriction base="string"/>
  </simpleType>
  <simpleType name="MapType" modelReference="
      http://...#RoadMap">
    <restriction base="anyURI"/>
  </simpleType>
  ...
</schema></types>
<message name="getMAPResponse">
  <part name="MapPart" type="MapType"/>
</message>
<message name="getMAPRequest">
  <part name="LocationPart" type="LocationType"/>
</message>
<portType name="LocationMapInterface">
  <operation name="createMAP">
    <input message="tns:getMAPRequest"/>
    <output message="tns:getMAPResponse"/>
  </operation>
  <operation ...>...</operation>
</portType>
```

```
<types><schema ...>
  <complexType name="GPSType" modelReference="
      http://...#GPSPos">
    <sequence>
      <element name="longitude" type="float"/>
      <element name="latitude" type="float"/>
    </sequence>
  </complexType>
  <simpleType name="MapType" modelReference="
      http://...#Map">
    <restriction base="anyURI"/>
  </simpleType>
</schema></types>
<message name="getMapResponse">
  <part name="_Map" type="MapType"/>
</message>
<message name="getMapRequest">
  <part name="_Location" type="GPSType"/>
</message>
<portType name="PositionMapInterface">
  <operation name="getMAP">
    <input message="tns:getMapRequest"/>
    <output message="tns:getMapResponse"/>
  </operation>
```

Gambar 2.1 Dokumen WSDL pada Web Service

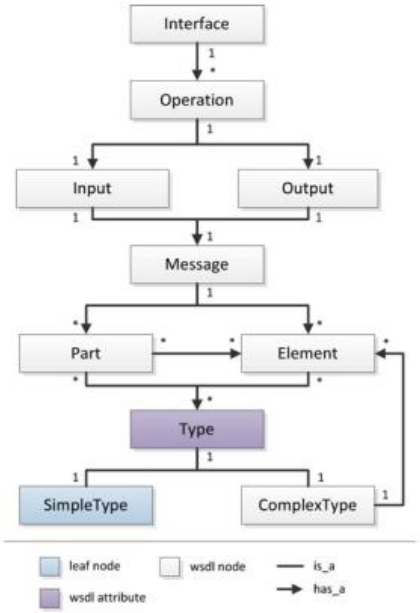
WSDL 1
Location map interface create map get map request get map response location part location type map party map type string any uri

WSDL 2
position map interface get map get map request get map response position gps type longitude latitude float float map type any uri

Gambar 2.2 Contoh hasil ekstraksi informasi pada WSDL

2.1.2 Kesamaan Struktur

Menurut Kamus Besar Bahasa Indonesia, struktur adalah pengaturan unsur atau cara sesuatu disusun atau dibangun. Web service dapat didefinisikan ke dalam bentuk dokumen WSDL. Struktur dari dokumen WSDL dibentuk seperti struktur XML. Gambar 2.3 menunjukkan ilustrasi struktur XML pada dokumen WSDL.



Gambar 2.3 Struktur XML pada WSDL

Berdasarkan Gambar 2.3, Unsur-unsur yang membangun struktur WSDL ada beberapa hal, yaitu operation, input, output, message, part, element dan type. Informasi struktur WSDL dapat digunakan sebagai aspek untuk mengukur kesamaan dari web service. Misalkan ada dua *file* WSDL seperti yang ditunjukkan pada Gambar 2.4, yaitu WSDL 1 dan WSDL 2. Setelah dibangun strukturnya berdasarkan skema pada Gambar 2.3 didapatkan hasil seperti yang ditunjukkan pada Gambar 2.5. Secara tekstual, WSDL 1 dan WSDL 2 memiliki kemiripan yaitu tentang buku, tetapi secara struktur WSDL 1 dan WSDL 2 memiliki input dan output yang berbeda sehingga fungsionalitas dari keduanya juga berbeda.

Pada pengukuran kesamaan semantik, web service diekstraksi informasi penting didalamnya. Hasil keluaran dari proses ekstraksi ini adalah kumpulan kata tak berurutan. Sehingga pada pengukuran kesamaan semantik, struktur dari Web Service diabaikan. Pengukuran kesamaan semantik tidak dapat membedakan mana input, output, message, elemen ataupun type. Ketika dilakukan proses pengukuran semantik, dua Web Service bisa saja memiliki makna tekstual yang sama tetapi input dan outputnya berbeda. Sehingga terjadi *false negative*, yaitu Web Service yang seharusnya tidak relevan dikatan relevan. Oleh Karena itu, pengukuran kesamaan struktur Web Service diperlukan dalam aktifitas penemuan Web Service.

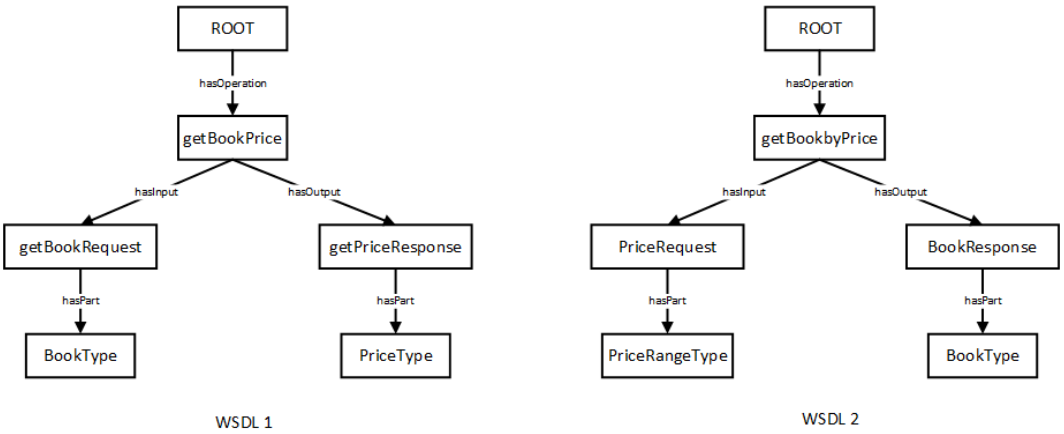
WSDL 1

```
<types><schema ...>
   <simpleType name="PriceType" modelReference="http://..#BookPrice">
       <restriction base="int"/>
    </simpleType>>
    <simpleType name="BookType" modelReference="http://..#Books">
       <restriction base="string"/>
    </simpleType>
    ...
</schema></types>

<message name="getPriceResponse">
    <part name="BookPrice" type="PriceType"/>
</message>
<message name="getBookRequest">
    <part name="BookName" type="BookType"/>
</message>

<portType name="getBookPriceInterface">
    <operation name="getBookPrice">
        <input message="tns:getBookRequest"/>
        <output message="tns:getPriceResponse"/>
    </operation>
<operation ...> ... </operation>
</portType>
```

WSDL 2

```
<types><schema ...>
   <complexType name="PriceRangeType" modelReference="http://..#BookPrice">
       <sequence>
          <element name="Price1" type="int"/>
          <element name="Price2" type="int"/>
       </sequence>
    </complexType>
    <simpleType name="BookType" modelReference="http://..#Books">
       <restriction base="BookList"/>
    </simpleType>
    ...
</schema></types>

<message name="getBookResponse">
    <part name="ListBook" type="BookType"/>
</message>
<message name="getPriceRequest">
    <part name="BookPrice" type="PriceRangeType"/>
</message>

<portType name="getBookbyPriceInterface">
    <operation name="getBookbyPrice">
        <input message="tns:getPriceRequest"/>
        <output message="tns:getBookResponse"/>
    </operation>
<operation ...> ... </operation>
</portType>
```

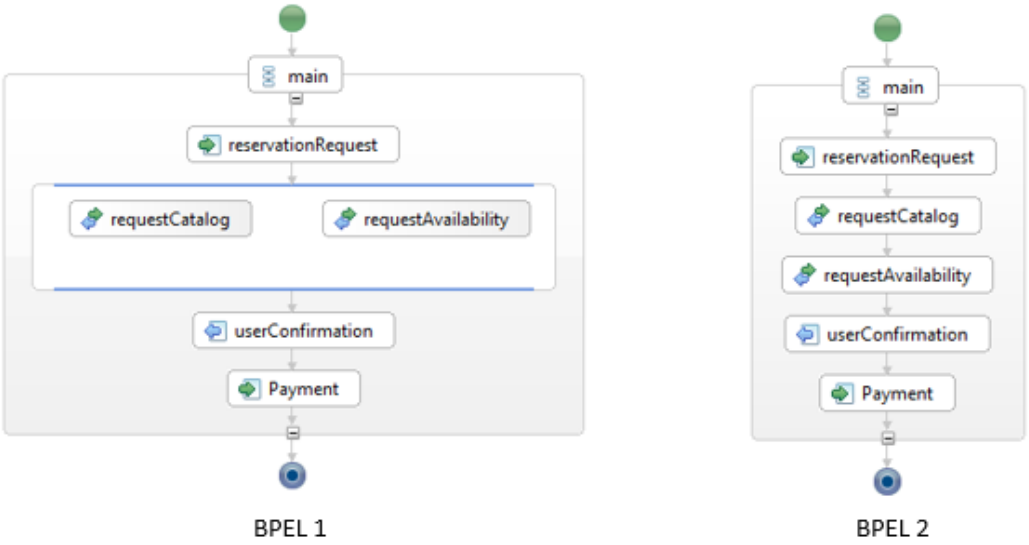
Gambar 2.4 Contoh dua *file* WSDL yang berbeda struktur



Gambar 2.5 Model struktur dari WSDL

2.1.3 Kesamaan Perilaku

Perilaku Web Service yaitu merupakan urutan sebuah service dalam memanggil operasi pada service lain untuk mencapai tujuannya (Wan, Mao, and Dai 2012). Web Service terbagi menjadi dua tipe yaitu Web Service atomik dan Web Service komposit. Web Service atomik adalah Web Service yang diimplementasikan hanya memanggil operasi yang ada didalamnya. Sedangkan Web Service komposit adalah Web Service yang diimplementasikan dengan memanggil service lain yang bukan bagian dirinya. Pengukuran kesamaan perilaku dilakukan untuk menemukan Web Service komposit yang mirip secara fungsional.



Gambar 2.6 Contoh Model BPEL untuk reservasi hotel

Perilaku Web Service dapat dimodelkan dengan Business Process Execution Language (BPEL). Gambar 2.6 menunjukkan dua web service komposit yang dimodelkan menjadi BPEL yaitu Web Service untuk reservasi hotel. Pada gambar tersebut dapat dilihat bagaimana urutan pemanggilan operasi sebuah web service komposit untuk memesan hotel pada BPEL 1, yaitu Pertama, melakukan reservasi dengan memanggil operasi ReservationRequest, Kemudian terdapat aktivitas Flow dimana dua proses didalam aktivitas Flow dapat dijalankan secara parallel. Selanjutnya operasi UserConfirmation dijalankan dan terakhir operasi Payment.

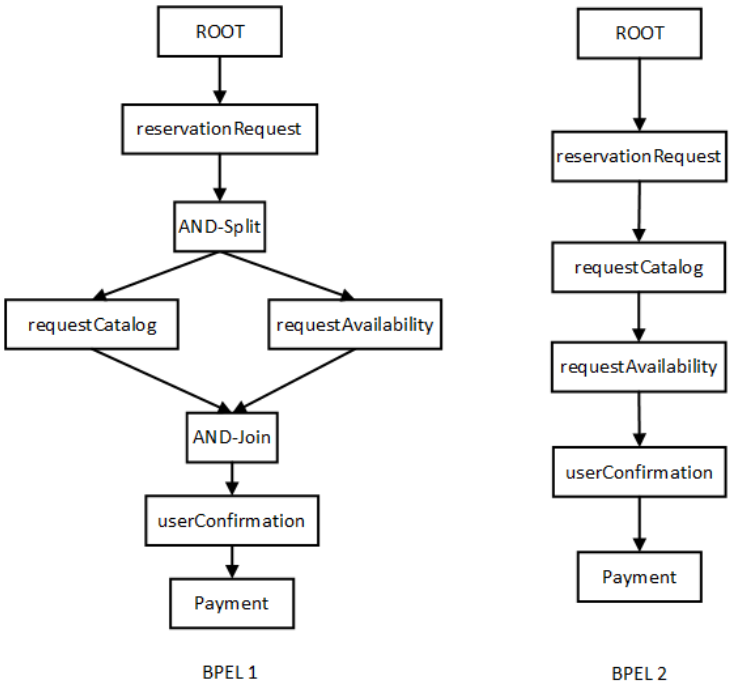
Urutan eksekusi operasi pada Web Service BPEL 1 adalah sebagai berikut:

(1) reservationRequest → requestAvailability → requestCatalog → userConfirmation → Payment

(2) reservationRequest → requestCatalog → requestAvailability → userConfirmation → Payment

Sedangkan untuk BPEL 2, urutan eksekusi operasinya adalah sebagai berikut:

(1) reservationRequest → requestCatalog → requestAvailability → userConfirmation → Payment



Gambar 2.7 Hasil Graph dari BCFTransformation

Secara struktur kedua BPEL tersebut berbeda karena pada BPEL 1 terdapat percabangan AND-Split dan AND-Join ketika dimodelkan menjadi graph menggunakan BCFTransformation (Grigori et al. 2010) seperti yang ditunjukkan oleh Gambar 2.7, tetapi memiliki kesamaan dalam eksekusi prosesnya. Hal ini dapat menyebabkan terjadinya *false positive*, dimana service yang seharusnya relevan justru dikatakan tidak relevan. Sehingga pengukuran kesamaan perilaku diperlukan untuk menemukan kemiripan antar Web Service komposit berdasarkan urutan pemanggilan operasinya.

2.2 Web Service

W3.org mendefinisikan Web Service sebagai "sebuah software aplikasi yang dapat teridentifikasi oleh URI dan memiliki interface yang didefiniskan, dideskripsikan, dan dimengerti oleh XML dan juga mendukung interaksi langsung dengan software aplikasi yang lain dengan menggunakan message berbasis XML melalui protokol internet".

Web Service sendiri dapat dijabarkan sebagai sebuah aplikasi lintas platform yang dapat diakses melalui jaringan (intranet dan internet). Dimana dalam

aplikasi tersebut menyediakan method – method dengan tujuan digunakan untuk interaksi aplikasi satu dengan aplikasi yang lain yang dapat diakses dengan URL dan menerima response berbentuk JSON, XML, TXT, CSV dan lainnya.

Tujuan utama penggunaan web service adalah Pelemparan data dari server satu dengan server lain yang berbeda lokasi (IP Address). Dengan mengakses URL kita bisa mendapatkan data dari aplikasi di environtment yang berbeda (maksud dari "environtment yang berbeda" adalah lokasi, sistem operasi, aplikasi, bahasa pemrograman). Web Service terbagi menjadi dua tipe yaitu web service atomik dan web service komposit. Web Service atomic dapat dideskripsikan dengan Web Service Definition Language (WSDL) dan Web Service komposit dapat dideskripsikan dengan Business Process Execution Language (BPEL).

2.2.1 Web Service Definition Language

Web Service Definition Language (WSDL) adalah format standar untuk menggambarkan layanan web. WSDL dikembangkan bersama oleh Microsoft dan IBM. WSDL membantu pengguna web service dalam memakai/menggunakan layanan web service. WSDL menspesifikasikan lokasi service dan operasi (methods) yang disediakan oleh web service.

WSDL sering digunakan pada kombinasi SOAP dan XML Schema untuk menyediakan Web service di internet. Aplikasi client yang menghubungkan ke sebuah Web service dapat membaca *file* WSDL untuk menentukan operasi apa saja yang tersedia pada server. Tipe-tipe data khusus yang digunakan di-embed pada *file* WSDL dalam bentuk XML Schema. Client kemudian dapat menggunakan SOAP untuk memanggil operasi-operasi yang terdaftar pada *file* WSDL secara aktual menggunakan XML atau HTTP.

Spesifikasi WSDL versi terkini adalah versi 2.0. Versi 1.1 belum didukung oleh W3C tetapi versi 2.0 sudah merupakan rekomendasi W3C. WSDL 1.2 diubah namanya menjadi WSDL 2.0 karena perbedaan substansial dari WSDL 1.1. Dengan mengijinkan binding ke semua metode HTTP request (tidak hanya GET dan POST seperti pada versi 1.1). Spesifikasi WSDL 2.0 menawarkan dukungan yang lebih baik untuk RESTful web service dan lebih sederhana atau simpel untuk diimplementasikan. Akan tetapi, dukungan untuk spesifikasi ini masih terlalu buruk

pada SDK untuk Web service yang sering menyediakan tools hanya untuk WSDL 1.1.

Elemen-elemen berikut yang terdapat dalam sebuah *File* WSDL, yaitu:

1. Message – sesuatu yang abstrak, definisi tipe data yang akan dikomunikasikan.
2. Port Type – mendeskripsikan sebuah web service, operasi-operasi yang dapat dijalankan, dan pesan-pesan yang dilibatkan pada Web Service.
3. Port - Titik akhir tunggal (single endpoint) yang didefinisikan sebagai sebuah 'binding' dan alamat jaringan (network address).
4. Service – Sekumpulan endpoint yang saling berhubungan, akan menunjukkan *file*/path mana yang akan ditempatkan pada *file* WSDL ini
5. Operation – deskripsi abstrak dari suatu aksi yang didukung oleh service. Pada dasarnya menunjukkan nama operasi web service dan pesan input output.
6. Binding – protokol komunikasi yang digunakan oleh web service.

2.2.2 Business Process Execution Language

BPEL merupakan sebuah bahasa yang mendeskripsikan proses eksekusi business process melalui web service. BPEL menggunakan XML untuk mendefinisikan komposisi fungsi-fungsi web service yang akan dieksekusi. BPEL adalah bahasa yang berbasis pada workflow yang menggabungkan service yang berperan dalam suatu interaksi yang dapat berupa orkestrasi dan koreografi service. BPEL memanfaatkan deskripsi WSDL untuk mendefinisikan fungsi yang disediakan oleh service yang ada dan model interaksi yang dibentuk.

BPEL merupakan kombinasi dari dua teknik terdahulu yang menangani mekanisme komposisi dari web service yaitu IBM Web Service Flow Language (WSFL) dan Microsoft XLANG (XLANG). BPEL menggabungkan dua teknik yang ada pada dua bahasa tersebut. Proses dalam BPEL dapat dibuat dengan menggunakan pemodelan yang graph-oriented yang disediakan oleh WSFL dan mekanisme pertukaran pesan yang ada pada XLANG.

Tujuan terkait dengan adanya BPEL yaitu adalah untuk medefinisikan proses bisnis yang berinteraksi dengan entitas luar melalui operasi web service yang di definisikan menggunakan WSDL 1.1 selain itu juga untuk mendefinisikan proses bisnis berdasarkan bahasa XML.

BPEL juga ditujukan untuk mendefinisikan satu set konsep orkestrasi web service yang akan digunakan oleh view eksternal (abstrak) dan internal (executable) dari bisnis proses, misalnya bisnis proses mendefinisikan tingkah laku dari single entity, cirinya beroperasi dengan cara saling berinteraksi antar entity yang sejenis. Hal ini mampu mengenali bahwa masing-masing penggunaan pola (misalnya abstrak view dan executable view) memerlukan sedikit ekstensi khusus, tetapi ekstensi ini akan tetap pada minimum dan terus diuji menggunakan permintaan seperti import/eksport dan membentuk pengecekan yang menyatukan kedua penggunaan pola tersebut.

Secara umum ada dua aktivitas dalam proses BPEL yaitu Basic Activities dan Structured Activities. BPEL menyediakan Basic Activities yang menggambarkan proses bisnis, antara lain Invoke, Reply, Receive, Throw dan Wait. Structured Activities menyediakan mekanisme untuk pembentukan logic dari workflow proses bisnis. Beberapa fungsi dalam aktivitas ini antara lain sequence, flow, switch, dan while.

2.3 Business Control Flow Transformation

Business Control Flow Transformation (BCFtransformation) merupakan metode untuk mengubah dokumen BPEL menjadi graph BPEL (Grigori et al. 2010). Aktivitas-aktivitas pada BPEL dimodelkan ke dalam bentuk graph. Tabel 2.2 menjabarkan rule yang digunakan untuk membuat BPEL graph. Aktivitas yang tidak ada di dalam tabel artinya tidak dihiraukan pada metode ini.

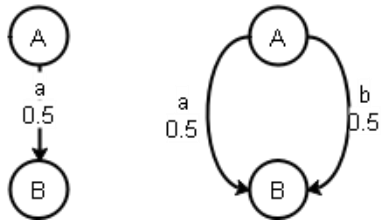
Tabel 2.2 Rule pembuatan BPEL Graph.

Activity name	**Rule**
Receive, Reply, Invoke, Wait	Dijadikan *Node*.

Sequence	Menghubungkan semua *node* pada sequence dengan arc.
While	Menghubungkan semua *node* pada while dengan arc.
Flow	Membuat cabang parallel pada AND-split dan kemudian disinkronisasi dengan AND-join.
Switch / Pick	Membuat cabang alternatif antara XOR-split dan XOR-join.

2.4 Weighted Directed Acyclic Graph

Weighted Directed Acyclic Graph (WDAG) didefinisikan sebagai sebuah graph G yang mempunyai *node* berlabel, arc berlabel, dan arc berbobot yang dikonstruksi dari 6 Tuple (N, V, L_N, L_V, L_W, r) yang terdiri dari kumpulan *node* N, kumpulan arc V, kumpulan label *node* L_N, kumpulan label arc L_V, dan kumpulan bobot arc $L_W = [0,1]$ dan element $r \in N$. Acyclic graph artinya tidak ada cycle atau loop pada graph.



Gambar 2.8 Contoh Weighted Directed Acyclic Graph

Contoh model Weighted Directed Acyclic Graph dapat dilihat pada Gambar 2.8. Pada penelitian ini, WDAG digunakan untuk memodelkan struktur dari dokumen WSDL dan juga memodelkan urutan eksekusi proses dari BPEL.

2.5 WDAG Similarity

Weighted Directed Acyclic Diagram Similarity (WDAG Similairty) (Jin 2006) adalah metode untuk mengukur kesamaan antar 2 Weighted Directed Acyclic Graph (WDAG), Algoritma WDAG Similarity dimulai dari menelusuri *node* root paling atas antara 2 WDAG yang memiliki label yang sama dan kemudian menelusuri *node* anak-anaknya dengan cara left-right depth first. Perhitungan kesamaan dimulai ketika kedua *node* yang sama berada sebagai *node* daun. Apabila kedua *node* tersebut memiliki label yang sama, maka WDAG Similarity = 1, apabila berbeda WDAG Similarity = 0. Kemudian nilai similaritas ini digunakan untuk menghitung nilai similaritas *node* parent nya.

Persamaan untuk menghitung similaritas antar *node* menggunakan WDAG Similarity dapat dilihat pada persamaan (2.1). Jika ada arc yang hilang pada salah satu (sub)WDAG yang ditelusuri, maka $wDAGsim(\varepsilon, g'_j) = 0.5 \cdot wDAGplicity(g'_j)$ atau $wDAGsim(g_i, \varepsilon) = 0.5 \cdot wDAGplicity(g_i)$ dan similaritasnya dihitung menggunakan WDAGplicity seperti pada persamaan (2.2). $wDAGsim(g, g')$:

$$\begin{cases} 0.0, \ root\ node\ label\ g\ dan\ g'\ tidak\ sama \\ 1.0, \ g\ dan\ g'\ adalah\ node\ daun \\ \sum \begin{cases} wDAGsim(g_i, g'_j) \cdot \frac{(w_i + w'_j)}{2}, \ apabila\ node\ g_i\ dan\ g_j\ tidak\ hilang \\ wDAGsim(g_i, \varepsilon) \cdot \frac{(w_i + 0)}{2}, \ apabila\ node\ g_i\ menghilang\ di\ g'_j \\ wDAGsim(\varepsilon, g'_j) \cdot \frac{(0 + w'_j)}{2}, \ apabila\ node\ g'_j\ menghilang\ di\ g_i \end{cases} \\ \sum_{j=1}^{breadth\_of\_g'} wDAGsim(\varepsilon, g'_j) \cdot \frac{(0 + w'_j)}{2}, \ apabil\ node\ g\ adalah\ node\ daun \\ \sum_{i=1}^{breadth\_of\_g} wDAGsim(g_i, \varepsilon) \cdot \frac{(w_i + 0)}{2}, \ apabila\ node\ g'\ adalah\ node\ daun \end{cases} \quad (2.1)$$

Dimana:

- $wDAGsim(g, g')$ adalah kemiripan antara dua buah WDAG g dan g'.
- $wDAGsim(g_i, g'_j)$ adalah kemiripan antara sub WDAG ke-*i* dan ke-*j*.
- w_i dan w'_j adalah bobot dari arc antara WDAG g_i dan WDAG g'_j yang dibandingkan. Sedangkan ε adalah WDAG kosong.

- *i* adalah nilai index yang meningkat dari 1 hingga banyaknya jumlah *node* anak dari WDAG g.
- *j* adalah nilai index yang meningkat dari 1 hingga banyaknya jumlah *node* anak dari WDAG g'.

$$wDAGplicity(g)=\begin{cases} D_I, & if\ g\ is\ leaf\ node \\ \frac{D_f}{m}\sum_{j=1}^{m} w_j \bullet wDAGplicity(g_j) \end{cases} \quad (2.2)$$

Dimana:

- D_{I} adalah Depth Degradation Index, sama dengan 1.
- D_f adalah Depth Degradation Factor, sama dengan 0.5
- m adalah banyaknya jumlah *node* anak pada (sub)WDAG g.

2.6 RuleML

RuleML merupakan sebuah Bahasa standar untuk merepresentasikan "Knowledge" atau Rule. RuleML memberikan kemudahan untuk memodelkan, mentranslasi, mengeksekusi, mempublikasi, dan menyimpan Rule dalam bentuk XML.

Pada penelitian (Jin 2006), RuleML digunakan untuk menserialisasikan Weighted Directed Acyclic Graph. RuleML memiliki logical tag yang dapat merepresentasikan WDAG. Gambar 2.9 menunjukkan contoh representasi WDAG dalam RuleML. Tiap tag *atom* pada RuleML menserialisasi satu (sub)WDAG. Alamat rujukan untuk suatu (sub)WDAG diserialisasikan pada tag *oid* dan untuk mengetahui alamat yang dirujuk suatu (sub)WDAG dapat dilihat pada atribut *uri*. Tag *slot* tanpa bobot digunakan untuk menyimpan *in-degree*, dimana *in-degree* adalah jumlah arc yang merujuk (sub)WDAG tersebut. Tag *slot* berbobot digunakan untuk menserialisasi arc yang dimiliki (sub)WDAG tersebut. Bobot dari arc ditunjukkan dengan atribut *weight* pada tag *slot*.

Pada penelitian ini, RuleML akan digunakan sebagai metadata untuk menyimpan informasi tentang struktur dan perilaku Web Service yang direpresentasikan dalam bentuk WDAG. Kemudian, *File* RuleML dijadikan

sebagai masukan untuk mengukur kesamaan struktur dan *behaviour* antar Web Service dengan mengukur nilai kesamaan antar WDAG menggunakan metode WDAG Similarity. Kesamaan semantik antar node label yang ada pada WDAG diukur kesamaannya dengan menggunakan metode JS-Divergence berdasarakan sebaran nilai probabilitas topik yang didapatkan dari metode Biterm Topic Model.

```
<?xml version="1.0" encoding="UTF-8"?>
<RuleML xmlns="http://www.ruleml.org/1.0/xsd" xmlns:xsi="http://www.w3.org/2001/XMLSc
<Assert mapClosure="universal">
    <Atom>
        <oid><Ind uri="#Root"/></oid>
        <Rel>Root</Rel>
        <slot><Ind>in-degree</Ind><Data>0</Data></slot>
        <slot weight="1.0"><Ind>hasNextProcess</Ind><Ind uri="process0"/></slot>
    </Atom>
    <Atom>
        <oid><Ind uri="#process0"/></oid>
        <Rel>login</Rel>
        <slot><Ind>in-degree</Ind><Data>1</Data></slot>
        <slot weight="0.5"><Ind>hasNextProcess</Ind><Ind uri="process1"/></slot>
        <slot weight="0.5"><Ind>hasNextProcess</Ind><Ind uri="process2"/></slot>
    </Atom>
    <Atom>
        <oid><Ind uri="#process1"/></oid>
        <Rel>one way flight search</Rel>
        <slot><Ind>in-degree</Ind><Data>1</Data></slot>
        <slot weight="1.0"><Ind>hasNextProcess</Ind><Ind uri="process3"/></slot>
    </Atom>
    <Atom>
        <oid><Ind uri="#process2"/></oid>
        <Rel>two way flight search</Rel>
        <slot><Ind>in-degree</Ind><Data>1</Data></slot>
        <slot weight="1.0"><Ind>hasNextProcess</Ind><Ind uri="process3"/></slot>
    </Atom>
    <Atom>
        <oid><Ind uri="#process3"/></oid>
        <Rel>add reservation</Rel>
        <slot><Ind>in-degree</Ind><Data>1</Data></slot>
        <slot weight="1.0"><Ind>hasNextProcess</Ind><Ind uri="process4"/></slot>
    </Atom>
```

Gambar 2.9 Contoh Representasi RuleML dari WDAG

2.7 Biterm Topic Model

Bi-Term Topic Model (BTM) merupakan variasi dari metode Latent Dirichlet Allocation (Yan et al. 2013). Pada kebanyakan metode Topic Modelling, topik direpresentasikan sebagai kelompok kata yang saling berhubungan dimana hubungannya berdasarkan pola kemunculan kata. Pemodelan topik konvensional mengeksploitasi pola kemunculan kata untuk mengungkapkan struktur semantic latent pada korpus secara tersirat dengan memodelkan generasi kata pada setiap dokumen.

Pendekatan ini sangat sensitif terhadap panjang dokumen dikarenakan pola kemunculan kata pada dokumen pendek sangat jarang dan tidak dapat diandalkan. Sebagai gantinya, jika semua pola kemunculan kata pada korpus digabungkan, maka frekuensinya menjadi lebih stabil dan korelasi antar kata semakin jelas, berdasarkan ide dasar ini dikembangkanlah metode Bi-Term Topic Model.

Gambar 2.10 menunjukkan algoritma pada metode Biterm Topic Model. Tahapan pertama dalam metode BTM adalah ekstraksi biterm. Biterm adalah pasangan kata tak berurutan yang muncul pada dokumen. Sebagai contoh sebuah dokumen pendek dengan tiga kata, w1, w2, dan w3 akan membentuk tiga biterm yaitu (w1,w2), (w1,w3),(w2,w3).

Input: jumlah topik K, hyperparameter α, β, Biterm set B
Output: parameter θ dan φ

Inisialisasi topik secara random untuk semua biterm
for *iter* = 1 to N_{iter} do
    for $b \in B$ do
        cari z_b berdasarkan $P(z_i = k|z_{-i}, B)$
        update n_z, $n_{w_{i|z}}$, dan $n_{w_{j|z}}$
    end
end
hitung parameter θ dan φ

Gambar 2.10 Algoritma Biterm Topic Model

Inisialisasi topik secara random diberikan kepada setiap biterm sebagai langkah awal pada metode ini. Selanjutnya dilakukan estimasi parameter untuk metode BTM. Estimasi parameter adalah proses melakukan sampling untuk mendapatkan nilai distribusi probabilitas kata pada topik (φ) dan distribusi probabilitas topik pada dokumen (θ). Pada metode BTM ada beberapa parameter-parameter yang digunakan yang dapat dilihat pada Tabel 2.3.

Tabel 2.3 Parameter pada BTM

Nama	Tipe	Fungsi
α (alpha)	Input	Hyperparameter yang mempengaruhi sebaran topik.

β (beta)	Input	Hyperparameter yang mempengaruhi sebaran topik.
K / T	Input	Jumlah Topik yang digunakan.
W	Input	Jumlah banyak kata yang muncul.
φ (phi)	Output	Distribusi probabilitas kata pada topik.
θ (theta)	Output	Distribusi probabilitas topik pada dokumen.
z	Output	Penentuan topik.

Untuk melakukan estimasi parameter output dengan sampling distribusi probabilitas, BTM menggunakan Gibbs Sampling. Sampling dilakukan dengan cara iterasi. Untuk setiap iterasi, untuk masing-masing topik, akan dilakukan random sampling berdasarkan kondisi pada persamaan (2.3).

$$P(z_i = k|z_{-i}, B) \propto \left(n_{-i,k} + \alpha\right) \frac{\left(n_{-i,w_i,1|k}+\beta\right)\left(n_{-i,w_i,2|k}+\beta\right)}{\left(n_{-i,\cdot|k}+W\beta\right)^2} \quad (2.3)$$

Dimana B adalah kumpulan dari biterm, z_{-i} adalah topik untuk semua biterm kecuali b_i, $n_{-i,k}$ adalah jumlah biterm yang mempunyai topik k kecuali b_i, $n_{-i,w_i|k}$ adalah jumlah kata w yang mempunyai topik k kecuali b_i. Ilustrasi random sampling pada BTM dapat dilihat pada Tabel 2.4.

Tabel 2.4 Ilustrasi sampling pada BTM

Term	z (topic)	Topik 1	Topik 2	Random sampling	new z
apple	1	0.000986	0.001809917	0.001598066	2
sit	1	0.158168	0.223152651	0.001465843	1

Tabel 2.4 mempunyai kolom Topik 1 dan Topik 2. Kolom tersebut merupakan hasil hitungan dari kondisi $P(z_i = k|z_{-i}, B)$. Random sampling dilakukan dengan mengambil sample acak dari range 0 hingga nilai kondisi Topik 2 yaitu dari range 0 - 0.001809917. Contoh pada kata “apple”, kata tersebut mempunyai z (topic) = 1. Ketika dilakukan random sampling didapatkan nilai 0.001598066, nilai random tersebut masuk kedalam range kondisi topik 2, yaitu 0.000986 - 0.001809917. Sehingga pada iterasi ini, kata “apple” akan diberikan topik baru yaitu topik 2.

Setalah proses iterasi dengan Gibbs Sampling selesai, BTM menghitung distribusi probabilitas kata pada topik (φ) dan distribusi probabilitas topik pada dokumen (θ). Setiap kata pada korpus dihitung distribusi probabilitasnya terhadap topik dengan menggunakan persamaan (2.4). Setiap dokumen dihitung distribusi probabilitas topiknya menggunakan persamaan (2.5).

$$\varphi_{k,w} = \frac{n_{w|k}+\beta}{n_{\cdot|k}+W\beta} \quad (2.4)$$

$$\theta_k = \frac{n_k+\alpha}{N_B+K\alpha} \quad (2.5)$$

Dimana n_k adalah jumlah biterm pada topik, $n_{w|k}$ adalah jumlah kata w yang diberikan topik k , $n_{\cdot|k}$ adalah jumlah kata pada topik k.

Penelitian ini mengusulkan penggunaan BTM untuk mengekstraksi topik pada dokumen WSDL web service didasarkan pada observasi hasil ekstraksi informasi pada dataset SAWSDL-TC3. Metode BTM ditujukan untuk menggali topik pada dokumen pendek. Tabel 2.5 menunjukkan beberapa hasil ekstraksi informasi pada dataset. Dokumen hasil ekstraksi informasi pada WSDL memiliki jumlah kata kurang dari 100 dan dapat dikategorikan sebagai dokumen pendek. Sehingga diharapkan metode BTM dapat menggali topik pada dokumen WSDL web service lebih baik dibandingkan dengan LDA.

Tabel 2.5 Beberapa hasil ekstrasi informasi penting pada dataset SAWSDL-TC3

Luxuryhotel_Heidelburgservice.wdsl
get luxury hotel response luxury hotel luxury hotel type get luxury hotel request luxury hotel soap get luxury hotel get luxury hotel request get luxury hotel response luxury hotel id luxury hotel data luxury hotel accomodation rating accomodation rating id luxury hotel accomodation rating data (45)
Bankeraddress_CityBankservice.wsdl
banker address get banker address request get banker address response banker banker type address address type banker address soap gender gender id gender

data variant name geographical region has time interval duration duration id unit measure time measure magnitude organization size organization size id (44)
Comedyfilmactionfilm_service.wsdl
get comedy film action film request get comedy film action film response comedy film type action film type comedy film action film soap comedy film action film service action film type comedy film type title comedy film action film (40)

2.8 Jensen-Shanon Divergence

Jensen-Shannon (JS) Divergence (Steyvers and Griffiths 2007) merupakan sebuah metode untuk menghitung jarak dari dua distribusi probabilitas. Pada penelitian ini, JS Divergence digunakan untuk menghitung kemiripan antara distribusi probabilitas topik pada 2 label *node* WDAG. JS Divergence bersifat simetris, artinya $D_{JS}(q,d) = D_{JS}(d,q)$. Misalkan ada dua set distribusi probabilitas q[x] dan d[x], maka perhitungan JS Divergence dinyatakan dalam persamaan (2.6).

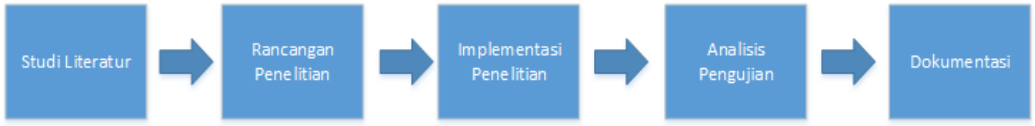
$$D_{JS}(q,d) = \frac{1}{2} D_{KL}\left(q, \frac{q+d}{2}\right) + \frac{1}{2} D_{KL}\left(d, \frac{q+d}{2}\right) \quad (2.6)$$

Dimana:

- $D_{KL}\left(q, \frac{q+d}{2} = r\right) = \sum_{i=1}^{T} q_i \ln \frac{q_i}{r_i}$.
- q dan d adalah dua buah distribusi probabilitas yang dibandingkan
- i adalah index yang meningkat dari 1 hingga banyaknya elemen dalam distribusi probabilitas topik.

BAB 3
METODE PENELITIAN

Bab ini akan memaparkan tentang metodologi penelitian yang digunakan pada penelitian ini, yang terdiri dari (1) studi literatur, (2) rancangan penelitian, (3) implementasi penelitian, (4) analisis pengujian, (5) dokumentasi sistem dan jadwal kegiatan. Ilustrasi alur metodologi penelitian dapat dilihat pada Gambar 3.1



Gambar 3.1 Alur Metodologi Penelitian

Penjelasan dari tahapan metode penelitian pada Gambar 3.1 akan diterangkan secara terperinci pada sub bab – sub bab berikutnya.

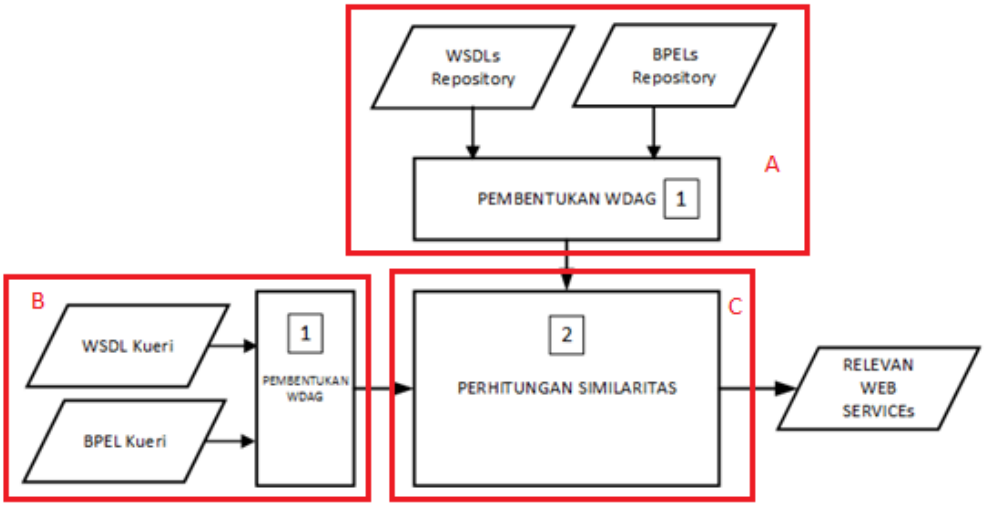
3.1 Studi Literatur

Penelitian selalu diawali dengan proses pengkajian yang berkaitan dengan topik penelitian yang diambil. Pada penelitian ini, referensi yang digunakan adalah jurnal – jurnal yang berkaitan dengan penemuan Web Service. Dari tahap ini, diharapkan dapat memberikan gambaran lengkap dan memberikan dasar kontribusi pembuatan kerangkat kerja untuk penemuan Web Service berdasarkan kesamaan semantik, struktural, dan *behavior*. Pada tahap ini studi literatur dilakukan secara sistematik (*Systematic Literature Review*). Literatur diambil dari jurnal dan konferensi internasional yang dipublikasi melalui portal sciencedirect, IEEE atau springer. Jurnal yang diambil dengan menggambil kosa-kata terkait penemuan Web Service berdasarkan kesamaan semantik, struktur, dan *behavior*.

3.2 Rancangan Penelitian

Kerangka kerja metode yang diusulkan pada penelitian ini terbagi atas 2 fase yaitu: (1) Pembuatan WDAG; (2) Perhitungan Similaritas. Alur rancangan kerangka kerja metode yang diusulkan pada penelitian ini dapat dilihat pada

Gambar 3.2. Fase 1, yaitu pembuatan WDAG, dilakukan untuk membentuk struktur dari WSDL dan *behavior* dari BPEL yang kemudian direpresentasikan ke dalam bentuk WDAG. Kemudian fase 2, yaitu perhitungan similaritas, dilakukan untuk menghitung nilai similaritas antara WDAG kueri dan WDAG yang ada direpositori. Perhitungan nilai similaritas antar WDAG dilakukan dengan menggunakan metode WDAG Similarity dengan menelusuri tiap *node* pada kedua WDAG yang memiliki *node* label yang mirip secara semantik. Kesamaan semantik antar label *node* WDAG yang ditelusuri dihitung berdasarkan nilai distribusi probabilitas topik yang didapatkan dari metode BTM.



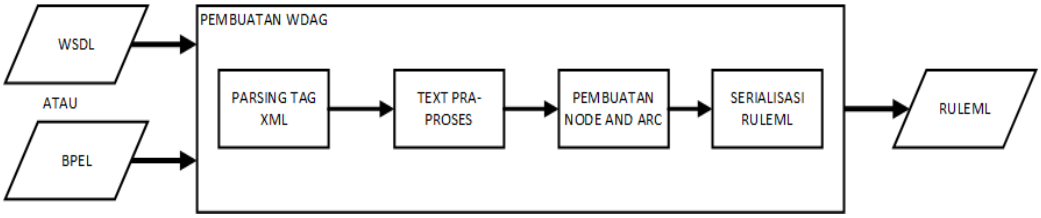
Gambar 3.2 Rancangan Metode Usulan

Alur proses yang pertama kali dilakukan pada rancangan metode usulan adalah membentuk WDAG dari WSDL dan BPEL yang ada direpositori seperti yang ditunjukkan oleh kotak merah berlabel A pada Gambar 3.2. Selanjutnya alur proses kedua adalah membentuk WDAG dari WSDL atau BPEL kueri seperti yang ditunjukkan oleh kotek merah berlabel B pada Gambar 3.2. Dan alur proses terakhir yaitu menghitung similaritas antara WDAG Kueri dan WDAG yang ada direpositori seperti yang ditunjukkan oleh kotak merah berlabel C pada Gambar 3.2.

3.2.1 Fase Pembuatan WDAG

Pada fase pembuatan WDAG, Web Service yang berupa WSDL atau BPEL dijadikan data masukan, kemudian dilakukan 4 proses untuk membangun

WDAG dari data masukan. Alur proses pembuatan WDAG dari dokumen WSDL dan BPEL ditunjukkan pada Gambar 3.3.



Gambar 3.3 Alur Pembuatan WDAG

a. Proses pertama yaitu Parsing Tag XML. Proses ini dibedakan menjadi dua proses sesuai dengan jenis masukan untuk WSDL dan BPEL.
   - Parsing Tag XML pada WSDL, yaitu proses mencari dan membaca tag XML pada *file* WSDL dan mengekstraksi attribute pada tag seperti "name", "type", dan "message". Contoh *file* WSDL yang dapat diparsing ditunjukkan pada Gambar 3.4.
   - Parsing Tag XML pada BPEL, yaitu proses mencari dan membaca tag XML pada *file* BPEL yang berupa elemen *activity* secara sekuensial. Attribute "name" pada tag *activity* seperti *sequence, flow, if,* dan lain-lain akan diekstraksi. Contoh *file* BPEL yang dapat diparsing ditunjukkan pada Gambar 3.5.

b. Proses kedua yaitu Text pra-proses. Proses ini dilakukan untuk mengolah teks yang didapat dari hasil proses parsing tag XML. Teks yang sudah diolah nantinya akan digunakan sebagai label pada *node* WDAG. Beberapa metode pengolahan teks digunakan seperti, tokenisasi, eliminasi stop word, dan lemmatisasi. Tokenisasi adalah proses untuk mengubah kalimat menjadi token-token. Rule diberikan untuk proses tokenisasi seperti dapat dilihat pada Tabel 3.1. Selanjutnya dilakukan proses eliminasi *Stopword*. *Stopword* adalah kata umum yang biasanya muncul dalam jumlah besar dan dianggap tidak memiliki makna. Contoh *Stopword* pada bahasa inggris diantaranya "of", "the", "a", "and", "an". Lemmatisasi adalah mengubah kata menjadi kata dasar seperti misalkan kata running menjadi kata run.

```
            <xsd:restriction base="xsd:string"/>
        </xsd:simpleType>
        <xsd:simpleType name="Once" sawsdl:modelReference="http://127.0.0.1/ontology/books.owl#Once">
            <xsd:restriction base="xsd:string"/>
        </xsd:simpleType>
        <xsd:simpleType name="Author" sawsdl:modelReference="http://127.0.0.1/ontology/books.owl#Author">
            <xsd:restriction base="xsd:string"/>
        </xsd:simpleType>
        <xsd:simpleType name="Title" sawsdl:modelReference="http://127.0.0.1/ontology/books.owl#Title">
            <xsd:restriction base="xsd:string"/>
        </xsd:simpleType>
        <xsd:simpleType name="Publisher" sawsdl:modelReference="http://127.0.0.1/ontology/books.owl#Publisher">
            <xsd:restriction base="xsd:string"/>
        </xsd:simpleType>
        <xsd:simpleType name="Book-Type" sawsdl:modelReference="http://127.0.0.1/ontology/books.owl#Book-Type">
            <xsd:restriction base="xsd:string"/>
        </xsd:simpleType>
    </xsd:schema>
</wsdl:types>
<wsdl:message name="get_PriceResponse">
    <wsdl:part name="_Price" type="PriceType">
    </wsdl:part>
</wsdl:message>
<wsdl:message name="get_PriceRequest">
    <wsdl:part name="_Book" type="BookType">
    </wsdl:part>
</wsdl:message>
<wsdl:portType name="BookPriceSoap">
    <wsdl:operation name="get_Price">
        <wsdl:input message="get_PriceRequest">
        </wsdl:input>
        <wsdl:output message="get_PriceResponse">
        </wsdl:output>
    </wsdl:operation>
</wsdl:portType>
```

Gambar 3.4 Contoh XML *File* WSDL

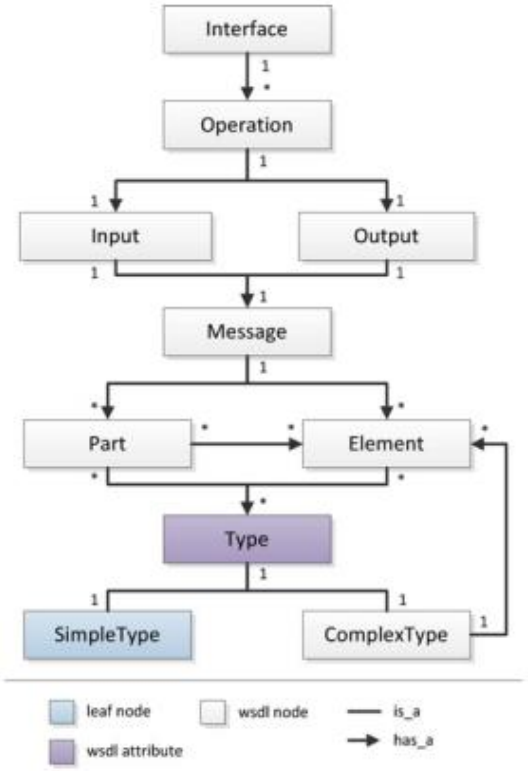
```
<bpel:sequence name="main">

    <!-- Receive input from requester.
         Note: This maps to operation defined in FlightReservation01.wsdl
         -->
    <bpel:invoke name="Login" partnerLink="clientPL" operation="Login" portType="tns:Flig


    <!-- Generate reply to synchronous request -->
    <bpel:if name="FlightSearch">
        <bpel:condition expressionLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.
        <bpel:invoke name="OneWayFlightSearch" partnerLink="FlightSearchPL" operation="On
    <bpel:elseif>
            <bpel:condition expressionLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpa
            <bpel:invoke name="TwoWayFlightSearch" partnerLink="FlightSearchPL" operation
        </bpel:elseif></bpel:if>

    <bpel:invoke name="AddReservation" partnerLink="FlightSearchPL" operation="OneWayFlig
    <bpel:reply name="Confirmation" partnerLink="client" operation="process" portType="tn
    <bpel:if name="Confirmation">
        <bpel:condition expressionLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.
            <![CDATA[$input.payload/tns:input="CancelReservation"]]>
        </bpel:condition>
        <bpel:invoke name="CancelReservation" partnerLink="clientPL" operation="process"
        <bpel:else>
            <bpel:sequence>
                <bpel:flow name="Flow">
            <bpel:invoke name="Payment" partnerLink="PaymentPL" operation="PaymentCC" por


        </bpel:flow>
                <bpel:invoke name="TicketEdit" partnerLink="clientPL" operation="TicketEd
            </bpel:sequence>
        </bpel:else>
    </bpel:if>

</bpel:sequence>
```

Gambar 3.5 Contoh XML *file* BPEL

Tabel 3.1 Rule untuk Tokenisasi

Rule	Original term	Versi Tokenisasi
Perubahan besar huruf 1	getBookRequest	get, book, request
Perubahan besar huruf 2	GetPrice	get, price
Eliminasi penomoran	Hello1	hello
Operator pemisah (e.g '_', '-')	Book-Type, Price_Type	book, type, price, type

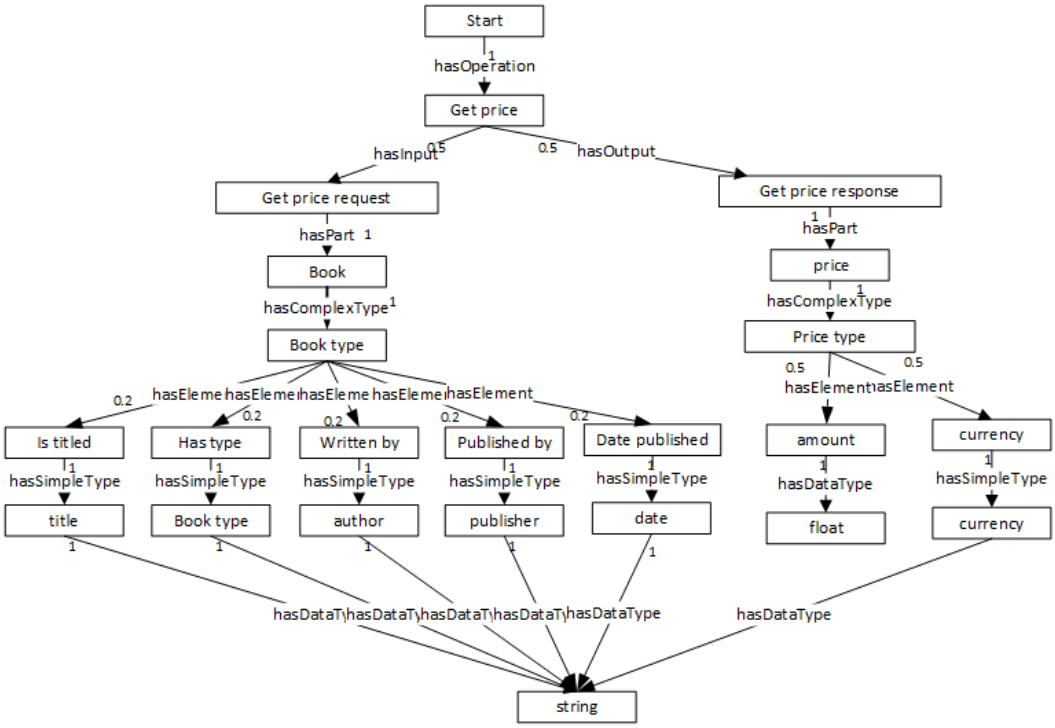
c. Proses ketiga yaitu Pembuatan *Node* dan Arc pada WDAG. Pembuatan *node* dan arc pada proses ini dibedakan menjadi dua proses berdasarkan jenis masukan yaitu WSDL atau BPEL. Apabila masukan berupa WSDL maka proses pembuatan *node* dan arc mengikut skema pada Gambar 3.6. Apabila masukan berupa BPEL maka proses pembuatan *node* dan arc mengikut rule BCFTransformation seperti pada Tabel 3.2. Contoh hasil dari pembuatan *node* dan arc pada *file* WSDL ditunjukkan pada Gambar 3.7 dan contoh hasil pembuatan *node* dan arc dari *file* BPEL ditunjukkan pada Gambar 3.8.
d. Proses terakhir yaitu Serialisasi RuleML. Pada proses ini *node* dan arc yang sudah dibuat pada proses sebelumnya disimpan sebagai metadata dalam *file* RuleML.



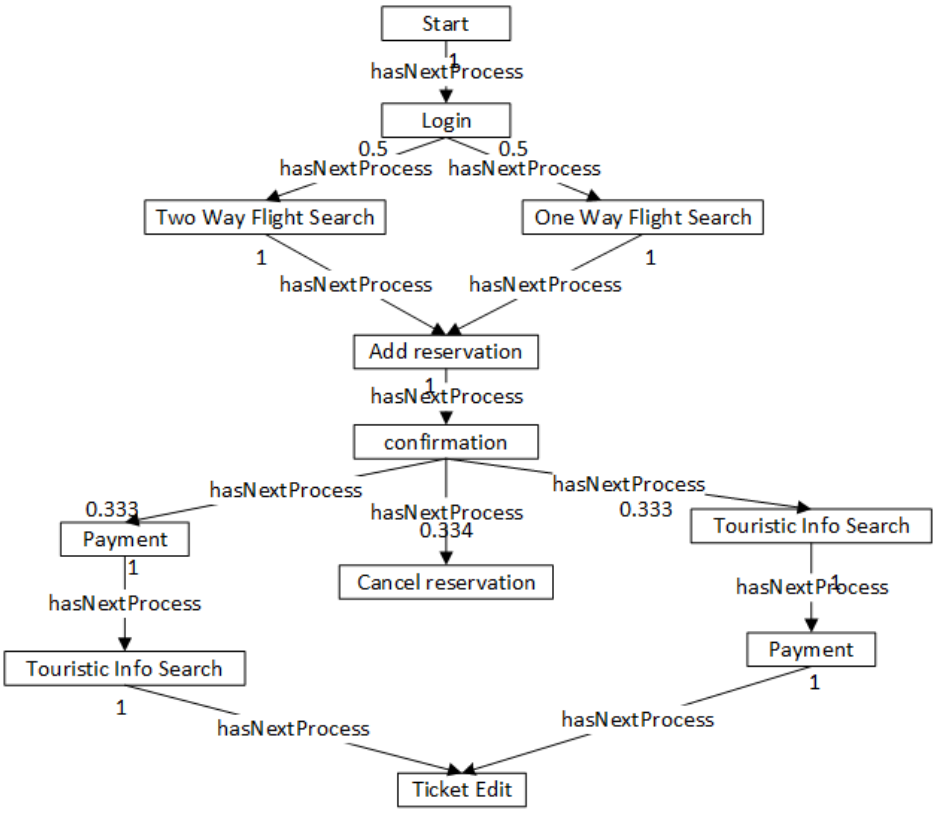
Gambar 3.6 Skema Pembentukan WDAG untuk WSDL

Tabel 3.2 Rule Pembentukan WDAG untuk BPEL

Activity name	Rule
Receive, Reply, Invoke, Wait	Dijadikan *Node*.
Sequence	Menghubungkan semua *node* pada sequence dengan arc.
While	Menghubungkan semua *node* pada while dengan arc.
Flow	Membuat cabang parallel pada AND-split dan kemudian disinkronisasi dengan AND-join.
Switch / Pick	Membuat cabang alternatif antara XOR-split dan XOR-join.



Gambar 3.7 Contoh WDAG WSDL

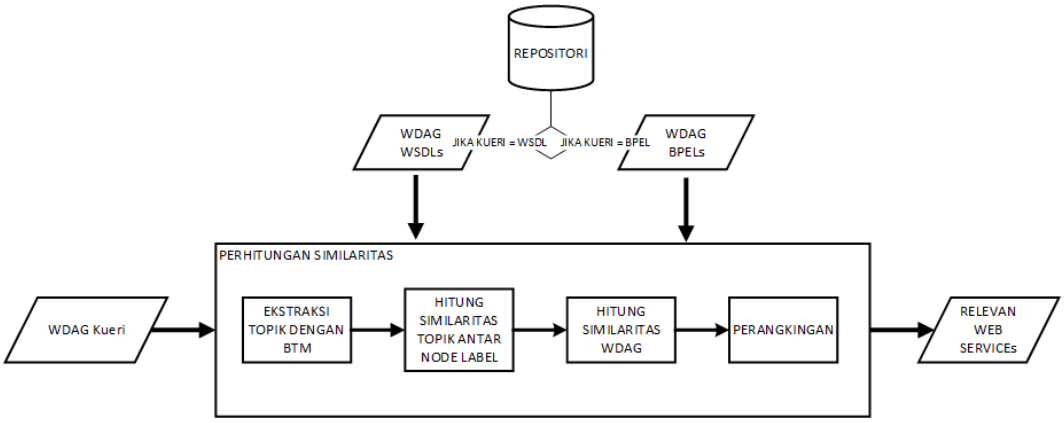


Gambar 3.8 Contoh WDAG BPEL

3.2.2 Fase Perhitungan Similaritas

Pada fase ini, WDAG kueri akan dihitung similaritasnya terhadap WDAG yang ada di repository. Alur dari fase perhitungan similaritas ini ditunjukkan pada Gambar 3.9. Apabila WDAG kueri adalah dari *file* WSDL maka *file* tersebut akan dibandingkan ke sesama *file* WDAG WSDL yang ada direpositori. Apabila WDAG kueri adalah dari *file* BPEL maka *file* tersebut akan dibandingkan ke sama *file* WDAG BPEL yang ada direpositori.

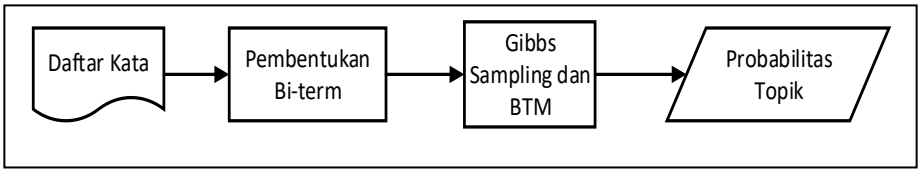
Selanjutnya, terdapat 4 proses dalam melakukan perhitungan similaritas antara WDAG kueri dan WDAG yang ada direpositori, yaitu (1) Ekstraksi Topik Dengan BTM, (2) Hitung Similaritas Topik antar *Node* Label, (3) Hitung Similaritas WDAG dan (4) Perangkingan.



Gambar 3.9 Alur Perhitungan Similaritas

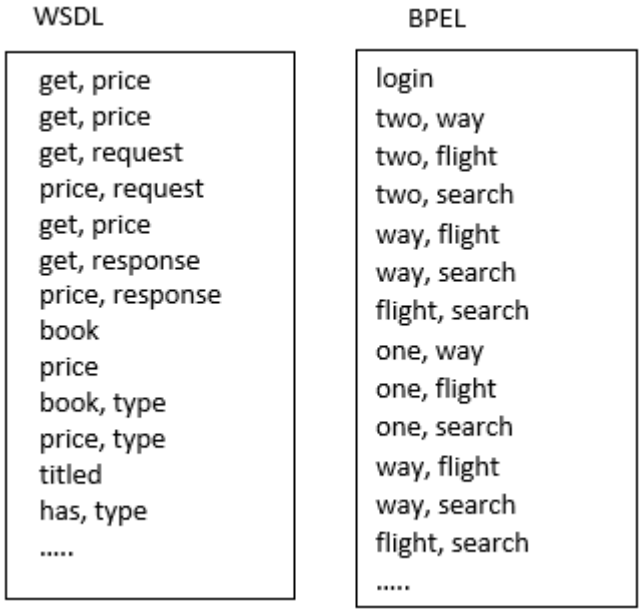
1. Ekstraksi Topik

Pada fase ini, metode BTM digunakan untuk mengekstraksi topik pada label *node* WDAG WSDL dan BPEL. Alur kerja untuk Fase Ekstraksi Topik dapat dilihat pada Gambar 3.10.



Gambar 3.10 Alur proses ekstraksi topik menggunakan BTM

Tahap pertama pada fase ekstraksi topik adalah dengan melakukan pembentukan bi-term untuk masing-masing *node* label pada WDAG WSDL dan BPEL. Cara pembentukan bi-term adalah dengan memasangkan satu kata dengan kata lainnya. Gambar 3.11 menunjukkan contoh hasil pembentukan korpus bi-term.



Gambar 3.11 Contoh Korpus bi-term yang sudah terbentuk

Kemudian korpus biterm ini digunakan pada metode BTM. Cara kerja BTM dan Gibbs Sampling sudah dijelaskan sebelumnya pada bab studi literatur. BTM menggunakan korpus biterm untuk menentukan probabilitas kata pada topik dimana proses sampling dilakukan dengan metode Gibbs Sampling. BTM akan menghasilkan dua keluaran, yaitu distribusi probabilitas kata pada topik (φ) dan distribusi probabilitas topik pada dokumen (θ). Masing-masing topik memiliki kata-kata dengan probabilitas kemunculannya. Ilustrasi hasil distribusi θ dengan jumlah topik = 30 dapat dilihat pada Tabel 3.3. Nilai distribusi θ ini yang nantinya akan digunakan sebagai masukan untuk mengukur similaritas distribusi topik pada label *node* WDAG.

Tabel 3.3 Ilustrasi hasil probabilitas topik di dokumen (θ)

Node Label	Topik 1	Topik 2	Topik 3	Topik 4	…..	Topik 30
get price	2.81E-10	4.06E-11	8.09E-10	4.65E-11	…..	6.76E-06
get price request	5.38E-10	7.77E-11	1.55E-09	8.90E-11	…..	0.0875319
get price response	0.1021	0.2001	0.3768	0.321	…..	0.0820828
car	0.4897	0.2541	0.1739	0.0823	…..	0.000261379
1 person bicycle	5.40E-10	7.80E-11	1.55E-09	8.93E-11	…..	0.000311656
price	0.00026176	0.992413	0.00026321	0.00026111	…..	4.05E-06
color	2.78E-07	4.63E-05	8.01E-07	4.60E-08	…..	0.000546898
engine	4.06E-06	4.05E-06	4.08E-06	4.05E-06	…..	0.0333059
person	0.000547694	0.0005463	0.000550728	0.000546335	…..	0.27821
wheel	0.0333544	0.0332695	0.0335392	0.0332716	…..	0.0333059
car type	7.33E-05	0.146301	7.37E-05	7.31E-05	…..	3.38E-05
shape	0.0333544	0.0332695	0.0335392	0.0332716	…..	0.0333059
power	6.00E-09	0.999768	1.73E-08	9.93E-10	…..	9.58E-05
life time	0.0333544	0.0332695	0.0335392	0.0332716	…..	3.48E-09
speed	9.60E-05	9.57E-05	9.65E-05	9.57E-05	…..	0.0333059
……	…..	…..	…..	…..	…..	…..
once	0.0333544	0.0332695	0.0335392	0.0332716	…..	0.0333059

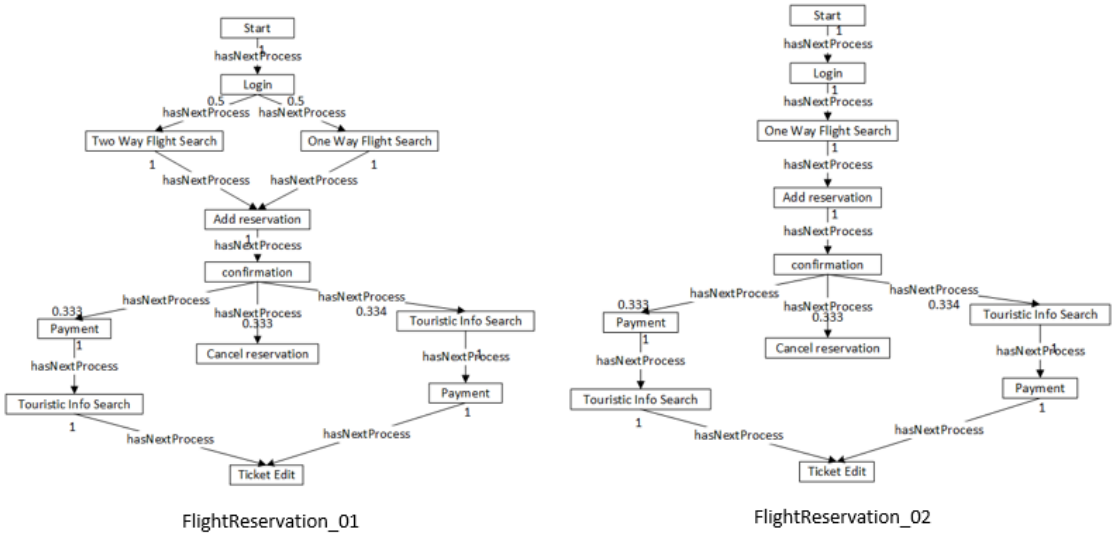
2. Hitung Similaritas Topik Antar Label

Pada fase ini, kemiripan semantik antara label *node* WDAG WSDL atau BPEL kueri dengan label *node* WDAG WSDL atau BPEL yang ada direpositori dihitung. Kemiripan semantic dihitung berdasarkan kesamaan distribusi probabilitas topik di dokumen (θ). Kemiripan distribusi topik pada dokumen dihitung menggunakan metode Jensen-Shanon Divergence. *Threshold* diberikan untuk menentukan kemiripan. Apabila hasil hitungan metode JS Divergence melebihi atau sama dengan *threshold*, maka kedua dokumen yang dibandingkan dapat dinyatakan memiliki kemiripan. Misal, kesamaan semantik antara label “get price” dengan “get price request” dihitung kesamaannya dengan nilai distribusi probabilitas topik yang

diambil dari Tabel 3.3. Hasil perhitungan similaritas dengan menggunakan metode JS Divergence pada persamaan (2.6) menghasilkan nilai 0.956596.

3. Hitung Similaritas WDAG

Pada fase ini, WDAG WSDL kueri dihitung kemiripannya dengan WDAG WSDL di repositori atau WDAG BPEL kueri dihitung kemiripannya dengan WDAG BPEL di repositori dengan menggunakan metode WDAG Similarity. Gambar 3.12 merupakan hasil pembuatan WDAG dari dua BPEL yaitu, FlightReservation_01 dan FlightReservation_02. WDAG Similarity melakukan penelusuran antara dua WDAG tersebut yang memiliki *node* label dan arc label yang sama untuk menghitung similaritas. Persamaan untuk menghitung similaritas antar WDAG dapat dilihat pada sub Bab 2.5. Apabila ada arc yang hilang pada salah satu *node* di WDAG yang dibandingkan, maka persamaan WDAGplicity digunakan untuk menghitung similaritas dari arc yang hilang.



Gambar 3.12 Contoh Komparasi 2 WDAG

Sebagai contoh, Gambar 3.12 menunjukkan dua BPEL WDAG yang akan dikomparasi. Metode WDAG Similarity dimulai dengan menelusuri anak *node* Root yang labelnya sama. Penelusuran dimulai pada *node* Login → One Way Flight Search → Add Reservation → Confirmation → Payment

→ Touristic Info Search → Ticket Edit. Karena *node* Ticket Edit adalah *leaf node* maka berdasarkan persamaan 2.1, WDAG Similarity pada *node* tersebut adalah 1. Kemudian perhitungan similaritas naik ke *parent node*, yaitu Touristic Info Search. WDAG Similarity pada *node* Touristic Info Search berdasarkan persamaan (2.1) adalah $1 \cdot \frac{(1+1)}{2} = 1$. Kemudian perhitungan similaritas naik ke *parent node* yaitu Payment. WDAG Similarity pada *node* Payment adalah $1 \cdot \frac{(1+1)}{2} = 1$. Kemudian perhitungan similaritas naik ke *parent node* yaitu Confirmation dengan *child node* Payment. WDAG Similarity pada *node* tersebut adalah $1 \cdot \frac{(0.333+0.333)}{2} = 0.333$.

Perhitungan similaritas dilanjutkan ke *child node* Confirmation kedua, yaitu Confirmation → Cancel Reservation. Karena *node* Cancel Reservation adalah *leaf node* maka berdasarkan persamaan (2.1), WDAG Similarity pada *node* tersebut adalah 1. WDAG Similarity pada *node* Confirmation dengan *child node* Cancel Reservation adalah $1 \cdot \frac{(0.333+0.333)}{2} = 0.333$.

Perhitungan similaritas dilanjutkan ke *child node* Confirmation ketiga, yaitu Confirmation → Touristic Info Search → Payment → Ticket Edit. WDAG Similarity pada *node* Ticket Edit adalah 1. Kemudian perhitungan dilanjutkan ke *parent node* yaitu Payment. WDAG Similarity pada *node* Payment adalah $1 \cdot \frac{(1+1)}{2} = 1$. Kemudian perhitungan dilanjutkan ke *parent node* yaitu Touristic Info Search. WDAG Similarity pada *node* Payment adalah $1 \cdot \frac{(1+1)}{2} = 1$. Kemudian perhitungan similaritas naik ke *parent node* yaitu Confirmation dengan *child node* Touristic Info Search. WDAG Similarity pada *node* tersebut adalah $1 \cdot \frac{(0.334+0.334)}{2} = 0.334$.

WDAG Similarity pada *node* Confirmation adalah total dari WDAG Similarity pada *child node* yaitu $0.333 + 0.334 + 0.333 = 1$. Selanjutnya perhitungan dilanjutkan ke *parent node* Add Reservation. WDAG Similarity pada *node* tersebut adalah $1 \cdot \frac{(1+1)}{2} = 1$. Selanjutnya perhitungan

dilanjutkan ke *parent node* One Way Flight Search. WDAG Similarity pada *node* tersebut adalah $1 \cdot \frac{(1+1)}{2} = 1$. Selanjutnya perhitungan dilanjutkan ke *parent node* Login. WDAG Similarity pada *node* Login dengan *child node* One Way Flight Search adalah $1 \cdot \frac{(0.5+1)}{2} = 0.75$.

Parent node Login pada FlightReservation_02 memiliki satu arc yang hilang pada FlightReservation_01, maka nilai $WDAG\ Similarity = (0.5 \cdot WDAGplicity)$. WDAGplicity dihitung menggunakan persamaan (2.2). WDAGplicity dihitung dengan menelusuri ulang *node* dengan label yang sama dari *node* yang hilang hingga ditemukan *leaf node*. Maka nilai WDAGplicity pada *node* Login dengan *child node* Two Way Flight Search adalah $\frac{0.5}{1} \cdot (0.0416458334 \cdot 1) = 0.0208229167$. Untuk mendapatkan nilai WDAG Similarity dari WDAGplicity maka dihitung sebagai berikut $0.5 \cdot 0.0208229167$ sehingga nilai similaritas pada *node* Two Way Flight Search yang hilang pada FlightReservation_02 adalah 0.01041145834. Selanjutnya dihitung WDAG Similarity *node* Login dengan *child node* Two Way Flight Search yaitu $0.01041145834 \cdot \frac{(0.5+0)}{2} = 0.002602865$ Sehingga total similaritas pada *node* Login adalah $0.75 + 0.002602865 = 0.752602865$.

Nilai similaritas dari dua WDAG pada Gambar 3.12 setelah dihitung dengan WDAG Similarity adalah sebesar 0.752602865. Nilai similaritasnya menurun dikarenakan terdapat satu arc yang hilang pada WDAG FlightReservation_02.

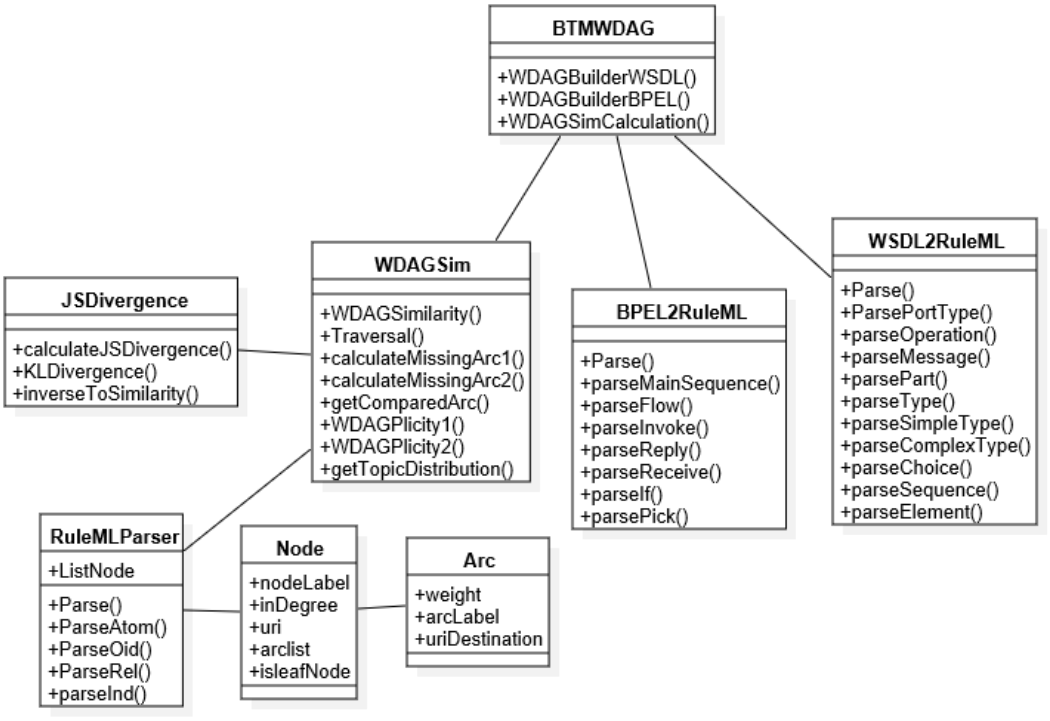
4. Perangkingan

Pada fase ini, hasil perhitungan similaritas WDAG akan diurutkan dari yang tertinggi hingga terendah. Nilai similaritas yang tinggi mengindikasikan bahwa Web Service tersebut yang ada di repositori memiliki kemiripan yang tinggi dengan Web Service kueri sedangkan semakin rendah nilai similaritasnya maka Web Service tersebut yang ada di repositori semakin berbeda dengan Web Service Kueri. *Threshold* diberikan untuk menentukan kemiripan. Apabila nilai kemiripan antara WDAG kueri

dengan WDAG repositori lebih dari atau sama dengan *threshold* maka WSDL atau BPEL yang bersangkutan yang ada di repositori dikembalikan sebagai Web Service yang relevan.

3.3 Implementasi Penelitian

Pendekatan penemuan Web Service yang diusulkan pada penelitian ini diimplementasikan dengan bahasa pemrograman Java. Untuk proses ekstraksi topik menggunakan BTM, penelitian ini menggunakan kakas bantu BTM-Master yang sudah dikembangkan oleh peneliti sebelumnya (Yan et al. 2013). Keluaran dari kakas bantu BTM-Master tersebut akan digunakan sebagai data untuk menghitung kesamaan semantik label *node* pada kakas bantu yang dibangun pada penelitian ini. Masukan yang diperlukan untuk kakas bantu pada penelitian ini adalah *file* WSDL dan BPEL sebagai kueri dan keluaran yang dihasilkan adalah daftar Web Service yang sudah diurutkan berdasarkan nilai similaritas dari WDAG Web Service kueri dan repositori.



Gambar 3.13 Diagram kelas dari kakas bantu yang diimplementasikan

Gambar 3.13 menunjukkan diagram kelas dari kakas bantu yang dibangun untuk membantu peneliti dalam menjalankan skenario pengujian yang sudah dirancang sebelumnya. Kelas utama sistem adalah BTMWDAG. Berdasarkan rancangan pada Gambar 3.2, fase pembuatan WDAG diimplementasikan menjadi Method WDAGBuilderWSDL() dan WDAGBuilderBPEL().

WDAGBuilderWSDL() dipanggil untuk membaca *file* WSDL masukan. Gambar 3.14 menunjukkan potongan kode sumber dari Method WDAGBuilderWSDL() yang diimplementasikan. Kemudian memanggil method Parse() pada kelas WSDL2RuleML. Potongan kode sumber dari methode Parse pada kelas WSDL2RuleML ditunjukkan oleh Gambar 3.15

```
public static void WDAGBuilderWSDL() throws IOException, JDOMException {
    String folder_path = "data-test/repository-wsdl/";
    final File folder = new File(folder_path);
    List<File> ListFile = FileUtils.listFilesForFolder(folder);
    String output_path = "data-test/ruleml-wsdl/";

    Collections.sort(ListFile, new Comparator<File>() {...8 lines });

    for(File file_name : ListFile) {
        String filePath = folder_path + file_name.getName();
        System.out.println("Processing: " + file_name.getName());
        //WSDL2RULEML.Parse(file_name.getName(), filePath, output_path);
        WSDL2RuleML2.Parse(file_name.getName(), filePath, output_path);
    }
}

public static void WDAGBuilderBPEL() throws IOException, JDOMException {
    String folder_path = "data-test/repository-bpel/";
    final File folder = new File(folder_path);
    List<File> ListFile = FileUtils.listFilesForFolder(folder);
    String output_path = "data-test/ruleml-bpel/";

    Collections.sort(ListFile, new Comparator<File>() {...8 lines });

    for(File file_name : ListFile) {
        String filePath = folder_path + file_name.getName();
        System.out.println("Processing: " + file_name.getName());
        //WSDL2RULEML.Parse(file_name.getName(), filePath, output_path);
        BPEL2RuleML2.Parse(file_name.getName(), filePath, output_path);
    }
}
```

Gambar 3.14 *Screenshot* Potongan Kode Sumber Method WDAGBuilder()

Kelas WSDL2RuleML melakukan parsing terhadap masukan WSDL dan menserialisasikan WDAG ke dalam RuleML. Method WDAGBuilderBPEL

membaca *file* BPEL masukan yang kemudian memanggil method Parse() pada kelas BPEL2RuleML seperti yang ditunjukkan pada Gambar 3.16. Kelas BPEL2RuleML melakukan parsing terhadap masukan BPEL dan menserialisasikan WDAG yang sudah dibangun ke dalam RuleML.

```
public static void Parse(String filename, String filepath, String output) throws IOException, JDOMException {
    ruleml_lines = new ArrayList<>();
    memorizeType = new HashMap<>();
    memorizeTypeDepth = new HashMap<>();
    memorizeTypeDepth = new HashMap<>();
    listPartType = new ArrayList<>();
    usedType = new ArrayList<>();
    TypeIndex = 0;
    documentRoot = null;
    fileName = filename;
    tokenizer = new MyTokenizer();

    output_path = output;

    File inputFile = new File(filepath);
    SAXBuilder saxBuilder = new SAXBuilder();
    Document document = saxBuilder.build(inputFile);
    documentRoot = document.getRootElement();
    parsePortType(documentRoot);

}
```

Gambar 3.15 *Screenshot* Potongan Kode Sumber Kelas WSDL2RuleML

```
public static void Parse(String filename, String filepath, String output) throws IOException, JDOMException {
    ruleml_lines = new ArrayList<>();
    memorizeProcess = new HashMap<>();
    memorizeProcessDepth = new HashMap<>();

    listPartType = new ArrayList<>();
    usedProcess = new ArrayList<>();
    ProcessIndex = 0;
    documentRoot = null;

    fileName = filename;
    tokenizer = new MyTokenizer();
    output_path = output;

    File inputFile = new File(filepath);
    SAXBuilder saxBuilder = new SAXBuilder();
    Document document = saxBuilder.build(inputFile);
    documentRoot = document.getRootElement();

    parseMainSequence(documentRoot);
    closeTagRuleML();
    FileUtils.WriteFile(output_path+ fileName+".ruleml", ruleml_lines);
}
```

Gambar 3.16 *Screenshot* Potongan Kode Sumber Kelas BPEL2RuleML

Fase kedua dari rancangan metode usulan yaitu fase perhitungan similaritas diimplementasikan menjadi Methode WDAGSimCalculation(). sMethod WDAGSimCalculation membaca dua *file* RuleML, yaitu *file* RuleML kueri dan *file* RuleML direpositori, yang dilakukan secara berulang sebanyak *file* RuleML direpositori, untuk dijadikan sebagai masukan yang akan dihitung

kesamaannya. Kemudian memanggil method WDAGSimilarity(). Gambar 3.17 menunjukkan potongan kode sumber dari kelas WDAGSim dengan Method WDAGSimilarity() yang telah diimpelementasikan.

Method WDAGSimilarity() kemudian memanggil method Parse() pada Kelas RuleMLParser untuk memparsing *file* RuleML masukan menjadi WDAG dan kemudian dilakukan penelusuran *node* dengan memanggil method Traversal(). Ketika membandingkan dua buah *node* label, maka method getTopicDistribution() dan method calculateJSDivergence() dipanggil.

```
public static double WDAGSimilarity(String file1, String file2, double threshold, String topicdisfilepath, String top
    try {
        WDAGSim.threshold = threshold;
        similarity = new HashMap<>();
        simplicity1 = new HashMap<>();
        simplicity2 = new HashMap<>();
        thisTemp1 = 0;
        thisTemp2 = 0;
        totalSimilarity = 0;
        topicPointerWDAG1 = new ArrayList<>();
        topicPointerWDAG2 = new ArrayList<>();

        getTopicDistribution(topicdisfilepath);
        getTopicPointer(file1, topicpointerpath, topicPointerWDAG1);
        //System.out.println("---------------------------------");
        getTopicPointer(file2, topicpointerpath, topicPointerWDAG2);

        List listNodeWDAG1 = RuleMLParser3.Parse(file1);
        List listNodeWDAG2 = RuleMLParser3.Parse(file2);
        Node2 rootNodeWDAG1 = getNodeByUri("#Root", listNodeWDAG1);
        Node2 rootNodeWDAG2 = getNodeByUri("#Root", listNodeWDAG2);
        File test2 = new File(file2);
        //System.out.print("with file2: " + test2.getName() + " = ");

        printString("\n######################\n");
        Traversal2(rootNodeWDAG1, rootNodeWDAG2, listNodeWDAG1, listNodeWDAG2);
        printString("finalSimilarity: " + totalSimilarity);
        //System.out.print(totalSimilarity + "\n");
        return totalSimilarity;
```

Gambar 3.17 *Screenshot* Potongan Kode Sumber Kelas WDAGSim

3.4 Rancangan Pengujian

Pada penelitian ini, *file* WSDL dan BPEL akan dipilih secara acak dan kemudian dijadikan kueri masukan untuk penemuan Web Service yang mirip yang ada direpositori dataset. Skenario pengujian untuk mengukur performa metode usulan yang akan dilakukan ada 3 macam skenario, yaitu:

1. Mengukur *Precision*, *Recall*, dan F-Measure dari metode usulan dengan mengubah-ubah banyaknya jumlah topik yang akan digali pada dataset.
2. Mengukur *Precision*, *Recall*, dan F-Measure dari metode usulan dengan mengubah-ubah jumlah *threshold* nilai kesamaan.

3. Membandingkan nilai *Precision*, *Recall* dan F-Measure dari metode usulan dengan metode lain dalam kasus penemuan Web Service.

BAB 4
PENGUJIAN DAN ANALISIS HASIL

4.1 Pengujian

Bab ini akan menjelaskan mengenai dataset dan alur pengujian yang dilakukan pada penelitian ini.

4.1.1 Dataset

Pengujian pada penelitian ini menggunakan 154 *file* WSDL dan 19 *file* BPEL. Data WSDL diambil dari Dataset SAWSDL-TC3 dan Data BPEL direkonstruksi dari (Grigori et al 2010) dan (Gunay and Yolum 2007). Data yang digunakan terdiri dari beberapa domain yang dapat dilihat pada Tabel 4.1. Daftar nama Web Service untuk setiap domain dapat dilihat di Bab Lampiran pada Tabel 7.1 - Tabel 7.8. Lima WSDL dan tiga BPEL dipilih secara acak dari dataset untuk digunakan sebagai kueri. Daftar kueri yang digunakan dapat dilihat pada Tabel 4.2.

Tabel 4.1 Kategori Dataset Pengujian

Domain	**Jumlah**
Car Web Service	18
Book Web Service	44
Film Web Service	58
Food Web Service	12
Academic Web Service	21
Government Web Service	8
Hotel Reservation Service	6
Flight Reservation Service	6
	173

Tabel 4.2 Daftar Kueri yang Digunakan Dalam Pengujian

Nama Service	**Tipe *file***	**Jumlah Relevan Service**
0_1personbicyclecar_price_service	WSDL	13
0_book_price_service	WSDL	16
0_bookpersoncreditcardaccount__service	WSDL	5
0_title_comedyfilm_service	WSDL	7
0_title_videomedia_service	WSDL	6
0_Book01	BPEL	3
0_FlightReservation01	BPEL	4
0_HotelReservation01	BPEL	3

4.1.2 Pengujian

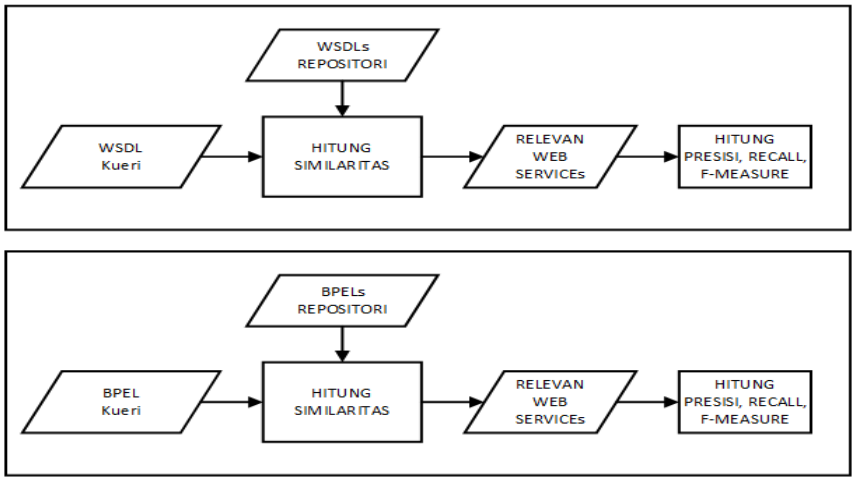
Analisis Pengujian yang dilakukan adalah dengan menggunakan metriks *Precision*, *Recall* dan F-measure. *Precision* dan *Recall* adalah dua perhitungan yang banyak digunakan untuk mengukur kinerja dari sistem atau metode yang digunakan pada bidang Information Retrieval. *Precision* adalah tingkat ketepatan antara informasi yang diminta oleh pengguna dengan jawaban yang diberikan oleh sistem. Sedangkan *Recall* adalah tingkat keberhasilan sistem dalam menemukan kembali sebuah informasi. *Precision* dan *Recall* dapat dihitung menggunakan persamaan (4.1) dan (4.2), dimana $\#(Relevant_i \wedge Retrieved_i)$ adalah jumlah jawaban yang relevan dengan permintaan user yang diberikan oleh sistem, $\#Retrieved_i$ adalah jumlah jawaban yang diberikan oleh sistem, dan $\#Relevant_i$ adalah jumlah jawaban yang relevan yang ada di dataset. F-measure adalah nilai harmonic *mean*s atau nilai keseimbangan dari *Precision* dan *Recall*. Nilai F-Measure dihitung dengan menggunakan persamaan (4.3)

$$Precision = \frac{\sum_i \#(Relevant_i \wedge Retrieved_i)}{\sum_i \#Retrieved_i} \% \quad (4.1)$$

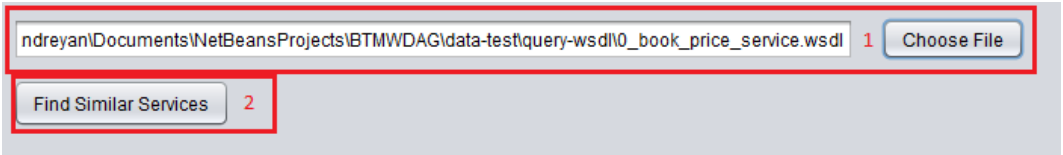
$$Recall = \frac{\sum_i \#(Relevant_i \wedge Retrieved_i)}{\sum_i \#Relevant_i} \% \quad (4.2)$$

$$F - Measure = 2 \times \frac{(Precision * Recall)}{(Precision + Recall)} \% \quad (4.3)$$

Gambar 4.1 menunjukkan alur pengujian yang dilakukan pada penelitian ini. satu WSDL ataupun BPEL digunakan sebagai kueri masukan. Apabila kueri adalah WSDL maka kueri tersebut akan dihitung similaritasnya dengan WSDL yang ada di repository. Apabila kueri adalah BPEL maka kueri tersebut akan dibandingkan dengan BPEL yang ada di repository. Similaritas antara kueri dan *file* yang ada di repository dihitung dan dikembalikan web service mana saja yang dianggap relevan dengan kueri. Kemudian hasil dievaluasi dengan menghitung *Precision*, *Recall*, dan F-measure dari hasil yang dikembalikan sistem dibandingkan dengan *groundtruth*.



Gambar 4.1 Alur pengujian



Gambar 4.2 Contoh Pengujian Memilih Kueri

No.	Nilai Similaritas	Nama Service
1.	1.000	2_book_Cheapestprice_service.wsdl.ruleml
2.	1.000	2_book_price_service.wsdl.ruleml
3.	1.000	2_book_recommendedprice_RegisteredUserservice.wsdl.ruleml
4.	1.000	2_book_recommendedprice_service.wsdl.ruleml
5.	1.000	2_book_recommendedpriceindollar_service.wsdl.ruleml
6.	1.000	2_BookPrice.wsdl.ruleml
7.	0.927	2_book_taxedprice_service.wsdl.ruleml
8.	0.891	2_book_reviewprice_service.wsdl.ruleml
9.	0.891	2_bookperson_price_service.wsdl.ruleml
10.	0.891	2_bookpersonOptional_price_service.wsdl.ruleml
11.	0.844	2_bookpersoncreditcardaccount_price_service.wsdl.ruleml
12.	0.844	2_bookpersoncreditcardaccount_recommendedprice_service.wsdl.ruleml
13.	0.844	2_bookpersoncreditcardaccount_taxedfreeprice_service.wsdl.ruleml
14.	0.834	2_book_pricereviewbook_service.wsdl.ruleml
15.	0.828	2_book_authorprice_Novelservice.wsdl.ruleml
16.	0.828	2_book_authorprice_service.wsdl.ruleml
17.	0.828	3_book_authorprice_Novelservice.wsdl.ruleml
18.	0.823	2_book_taxedpriceprice_service.wsdl.ruleml
19.	0.781	2_book_pricesizebook-type_service.wsdl.ruleml
20.	0.781	3_book_pricesizebook-type_service.wsdl.ruleml
21.	0.530	2_carbicycle_price_service.wsdl.ruleml
22.	0.530	2_carbicycle_recommendedprice_service.wsdl.ruleml

Gambar 4.3 Hasil Keluaran dari Sistem

Pada penelitian ini, alur pengujian dilakukan pertama kali dengan memilih *file* kueri seperti yang ditunjukkan pada Gambar 4.2. Kueri dapat berupa *file* WSDL atau BPEL. Sebagai contoh, *file* WSDL book_price_service.wsdl dipilih sebagai

kueri dan digunakan sebagai masukan pada sistem. Selanjutnya dilakukan perhitungan similaritas dengan mengklik tombol “Find Similar Services”. Terakhir, sistem menampilkan daftar Web Services di repositori beserta nilai similaritasnya terhadap kueri seperti yang ditunjukkan oleh Gambar 4.3. Hasil keluaran ini kemudian akan dibandingkan dengan *groundtruth* menggunakan program Excel untuk menghitung nilai *Precision* berdasarkan persamaan (4.1), *Recall* berdasarkan persamaan (4.2), dan F-Measure berdasarkan persamaan (4.3) seperti yang ditunjukkan oleh Gambar 4.4. Dari hasil perbandingan dengan *groundtruth*, kueri book_price_service.wsdl mendapatkan nilai *Precision* dan *Recall* tertinggi dengan nilai *threshold* similaritas 0.6 dan 0.7 sebesar 80% dan 100%.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
2_book_C	1	T			th 0.1	th 0.2	th 0.3	th 0.4	th 0.5	th 0.6	th 0.7	th 0.8	th 0.9			
2_book_p	1	T														
2_book_re	1	T		retrieved:	99	55	55	34	31	20	20	18	7			
2_book_re	1	T		relevan & retrieved:	16	16	16	16	16	16	16	14	7			
2_book_re	1	T		relevan:	16	16	16	16	16	16	16	16	16			
2_BookPri	1	T														
2_book_ta	0.927091	T		precision:	16.16162	29.09091	29.09091	47.05882	51.6129	80	80	77.77778	100			
2_book_re	0.890625	T		recall:	100	100	100	100	100	100	100	87.5	43.75			
2_bookpe	0.890625	T		F-Measure:	27.82609	45.07042	45.07042	64	68.08511	88.88889	88.88889	82.35294	60.86957			
2_bookpe	0.890625															
2_bookpe	0.843744			Kueri:	0_book_price_service.wsdl.ruleml											
2_bookpe	0.843744															
2_bookpe	0.843744															
2_book_p	0.833759	T			th 0.4	th 0.5	th 0.6	th 0.7	th 0.8	th 0.9						
2_book_a	0.82815	T														
2_book_a	0.82815	T		retrieved:	34	31	20	20	18	7						
3_book_a	0.82815	T		relevan retrieved:	16	16	16	16	14	7						
2_book_ta	0.823248	T		relevan:	16	16	16	16	16	16						
2_book_p	0.781269	T			precision						recall					
3_book_p	0.781269	T			47.05882	51.6129	80	80	77.77778	100	100	100	100	100	87.5	43.75
2_carbicyc	0.529991															
2_carbicyc	0.529991															
2_carcycle	0.529991															
2_car2per	0.522179															

Gambar 4.4 Hasil perbandingan dengan *groundtruth* menggunakan Excel

4.2 Analisis Hasil

Bab ini akan menjelaskan mengenai analisis hasil pengujian berdasarkan skenario pengujian yang sudah ditentukan.

4.2.1 Analisis Hasil Skenario Pengujian 1

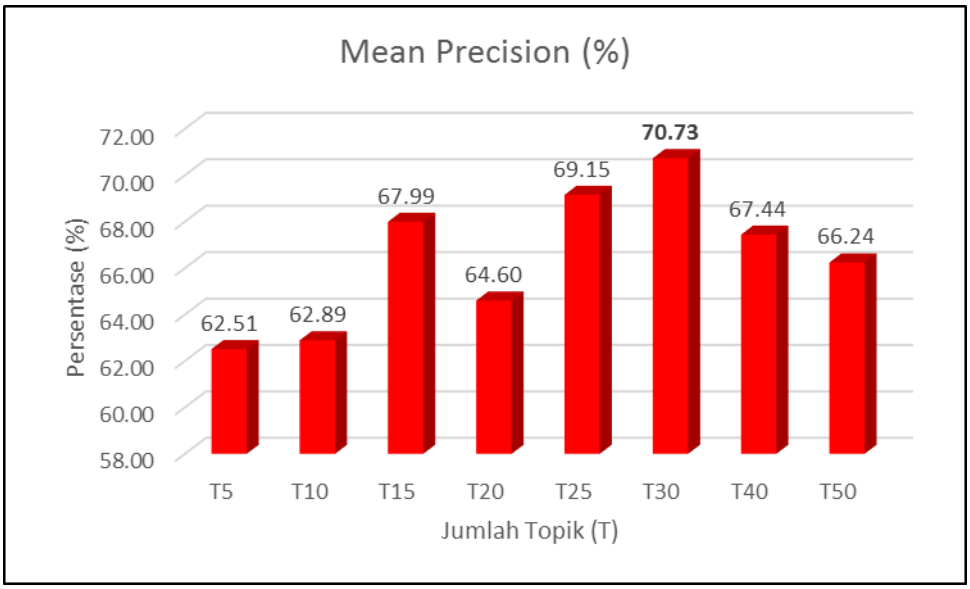
Dalam skenario 1, pengujian dilakukan dengan mengubah-ubah jumlah topik yang akan digali pada label *node* WDAG. Pengujian ini dilakukan dikarenakan jumlah topik dapat mempengaruhi distribusi probabilitas topik yang dihasilkan oleh metode BTM. Sehingga, hasil distribusi probabilitas topik tersebut dapat mempengaruhi pengukuran kesamaan label *node* pada WDAG.

Pengujian ini dilakukan untuk melihat tren nilai *Precision*, *Recall* dan F-measure ketika jumlah topik berbeda. Tabel 4.3 menunjukkan nilai *mean Precision* dari jumlah topik yang berbeda. Nilai *Precision* dihitung menggunakan persamaan (4.1). Nilai *Precision* terbaik terdapat pada jumlah topik = 30 dengan nilai 70.73%. Nilai *Precision* yang diperoleh lebih besar dari 50% dikarenakan rasio Web Service yang relevan lebih banyak, dibandingkan dengan rasio Web Service yang tidak relevan dari total jumlah Web Service yang dikembalikan sistem.

Tabel 4.3 Nilai *Mean Precision* pada Skenario Pengujian 1

Jumlah Topik	*Mean Precision* (%)
T5	62.51
T10	62.89
T15	67.99
T20	64.60
T25	69.15
T30	**70.73**
T40	67.44
T50	66.24

Gambar 4.5 menunjukkan tren dari nilai *Precision* masing-masing jumlah topik.

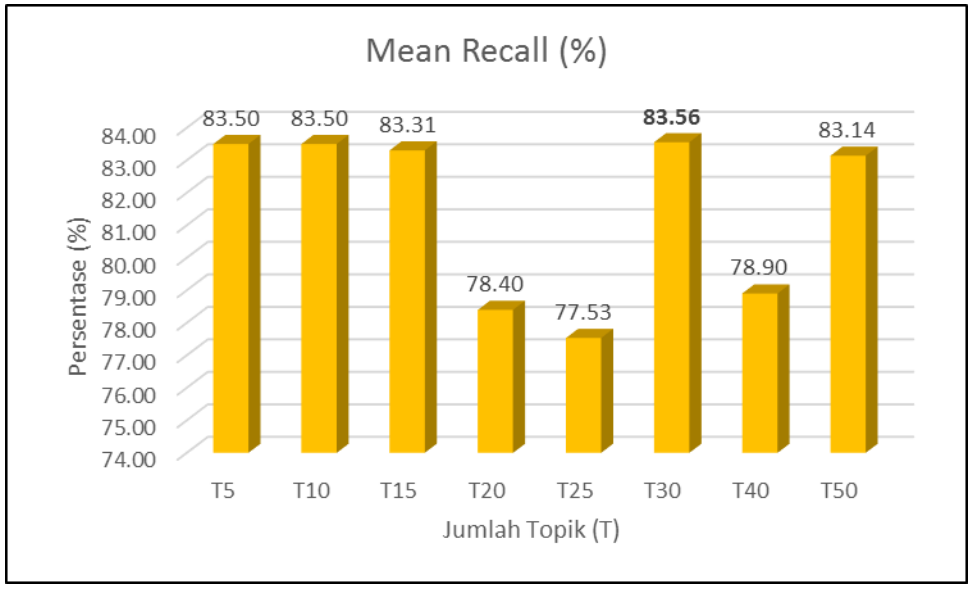


Gambar 4.5 Grafik Tren *Mean Precision* pada Skenario Uji 1

Tabel 4.4 menunjukkan nilai *mean Recall* dari jumlah topik yang berbeda. Nilai *Recall* dihitung menggunakan persamaan (4.2). Nilai *Recall* terbaik terdapat pada jumlah topik = 30 dengan nilai 83.56%. Nilai *Recall* yang didapatkan lebih besar dari 50% dikarenakan rasio jumlah Web Service yang relevan yang dikembalikan oleh sistem hampir sebanyak jumlah Web Service relevan yang ada di dataset. Tren dari nilai *Recall* dengan jumlah topik berbeda-beda ditunjukkan pada Gambar 4.6.

Tabel 4.4 Nilai *Mean Recall* pada Skenario Uji 1

Jumlah Topik	*Mean Recall* (%)
T5	83.50
T10	83.50
T15	83.31
T20	78.40
T25	77.53
T30	**83.56**
T40	78.90
T50	83.14



Gambar 4.6 Grafik Tren *Mean Recall* pada Skenario Uji 1

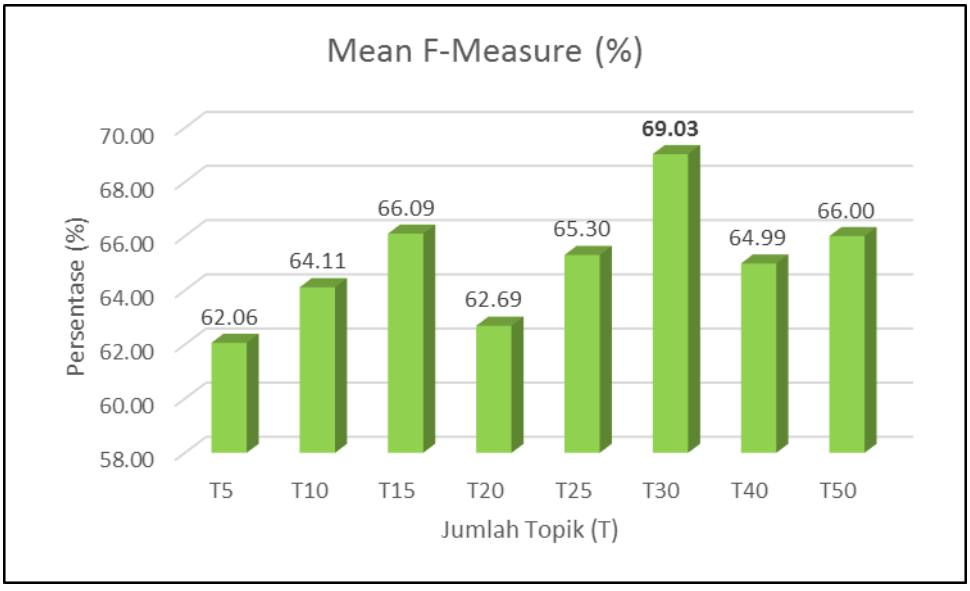
Untuk mengetahui jumlah topik mana yang mempunyai nilai *Precision* dan *Recall* terbaik secara keseluruhan, maka perlu dihitung nilai harmonic *mean*

antara *Precision* dan *Recall* yaitu dengan menghitung nilai F-Measure. Nilai F-Measure dihitung menggunakan persamaan (4.3). Nilai F-Measure menunjukkan rasio trade-off terbaik antara nilai *Precision* dan *Recall*.

Tabel 4.5 menunjukkan nilai *mean* F-Measure dari masing-masing jumlah topik. Nilai *mean* F-Measure terbaik terdapat pada jumlah topik = 30 dengan nilai 69.03%. Tren dari nilai F-Measure dengan jumlah topik berbeda-beda ditunjukkan oleh Gambar 4.7

Tabel 4.5 Nilai *Mean* F-Measure pada Skenario Uji 1

Jumlah Topik	*Mean* F-Measure (%)
T5	62.06
T10	64.11
T15	66.09
T20	62.69
T25	65.30
T30	**69.03**
T40	64.99
T50	66.00



Gambar 4.7 Grafik Tren *Mean* F-Measure pada Skenario Uji 1

Dari hasil pengujian skenario 1, jumlah topik sama dengan 30 mendapatkan nilai rata-rata *Precision*, *Recall*, dan F-Measure terbaik sebesar 70.73%, 83.56%, dan 69.03%. Dari hasil pengujian skenario 1 dapat disimpulkan

bahwa ketika mengukur kesamaan semantik dengan menggunakan metode pemodelan topik, jumlah topik dapat mempengaruhi nilai *Precision* dan *Recall*. Apabila terlalu sedikit menentukan jumlah topik maka dapat menyebabkan label *node* yang seharusnya berbeda topik justru memiliki topik yang sama. Apabila terlalu banyak menentukan jumlah topik maka dapat menyebabkan label *node* yang seharusnya memiliki topik yang sama justru memiliki topik yang berbeda.

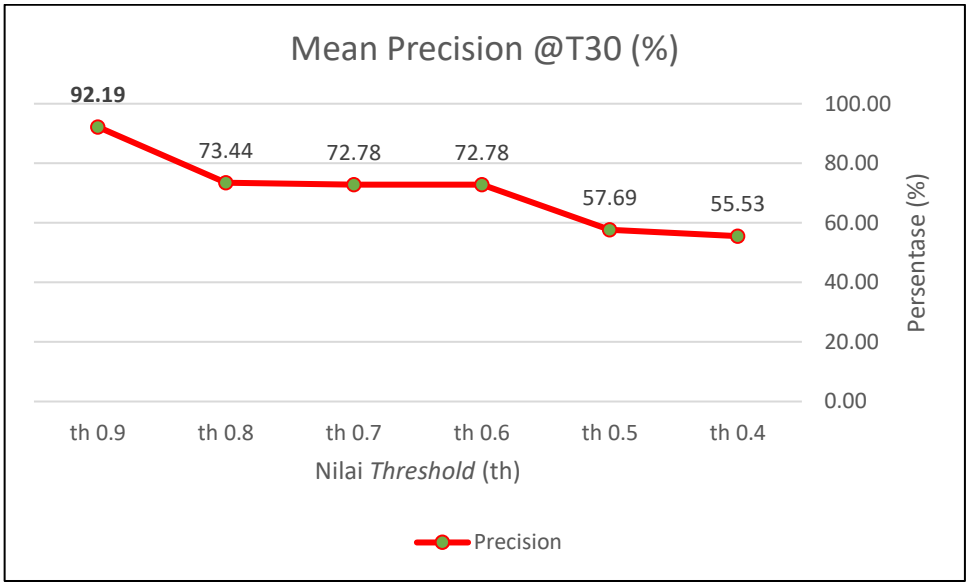
4.2.2 Analisis Hasil Skenario Pengujian 2

Dalam skenario 2, pengujian dilakukan dengan mengubah-ubah nilai *threshold* similaritas WDAG. Pengujian ini dilakukan dikarenakan nilai *threshold* mempengaruhi banyaknya jumlah *file* WSDL atau BPEL yang ada direpositori yang dikembalikan oleh sistem, sehingga hal ini akan mempengaruhi nilai *Precision*, *Recall*, dan F-Measure. Jumlah topik yang digunakan adalah T = 30, dimana jumlah topik ini menghasilkan *mean* F-Measure terbaik dari proses pengujian sebelumnya. Tabel 4.6 menunjukkan nilai *mean Precision* pada skenario pengujian 2.

Nilai *Precision* terbaik terdapat pada *threshold* 0.9 dengan nilai 92.19% dikarenakan rasio *file* WSDL atau BPEL yang relevan yang dikembalikan oleh sistem lebih banyak dari rasio yang tidak relevan dengan *threshold* 0.9. Tren dari nilai *mean Precision* dengan nilai *threshold* berbeda-beda ditunjukkan oleh Gambar 4.8. Semakin dikurangi nilai *threshold*-nya, maka nilai *Precision*nya juga semakin berkurang dikarenakan rasio *file* yang tidak relevan semakin banyak yang dikembalikan oleh sistem.

Tabel 4.6 Nilai *Mean Precision* pada Skenario Uji 2

Threshold	***Mean Precision* (%)**
th 0.9	**92.19**
th 0.8	73.44
th 0.7	72.78
th 0.6	72.78
th 0.5	57.69
th 0.4	55.53

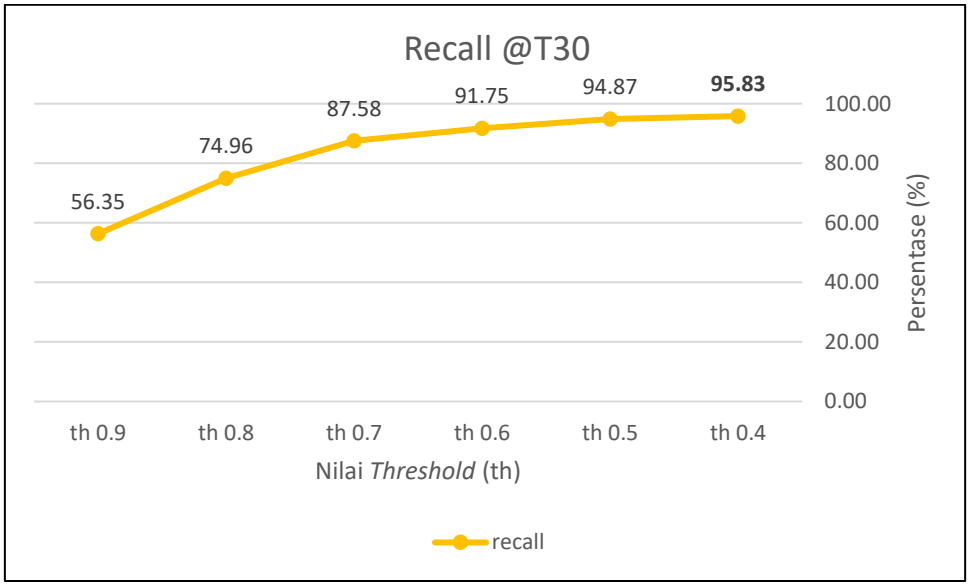


Gambar 4.8 Grafik Tren *Mean Precision* pada Skenario Uji 2

Tabel 4.7 menunjukkan nilai *mean Recall* pada skenario pengujian 2. Nilai *Recall* terbaik terdapat pada *threshold* 0.4 dengan nilai 95.83% dikarenakan rasio *file* WSDL atau BPEL yang relevan yang ada direpositori banyak yang dikembalikan oleh sistem dengan *threshold* 0.4. Tren dari nilai *mean Recall* dengan nilai *threshold* berbeda-beda ditunjukkan oleh Gambar 4.9. Nilai *Recall* semakin bertambah seiring berkurangnya nilai *threshold* dikarenakan rasio *file* WSDL atau BPEL yang relevan yang ada direpositori semakin banyak yang dikembalikan oleh sistem. Namun hal ini berdampak terbalik untuk nilai *Precision*. Sehingga untuk memutuskan nilai *threshold* mana yang menghasilkan nilai *Precision* dan *Recall* terbaik, perlu dihitung rasio trade-off terbaik antara *Precision* dan *Recall*, yaitu dengan menghitung nilai F-Measure.

Tabel 4.7 Nilai *Mean Recall* pada Skenario Uji 2

Threshold	***Mean Recall* (%)**
th 0.9	56.35
th 0.8	74.96
th 0.7	87.58
th 0.6	91.75
th 0.5	94.87
th 0.4	**95.83**



Gambar 4.9 Grafik Tren *Mean Recall* pada Skenario Uji 2

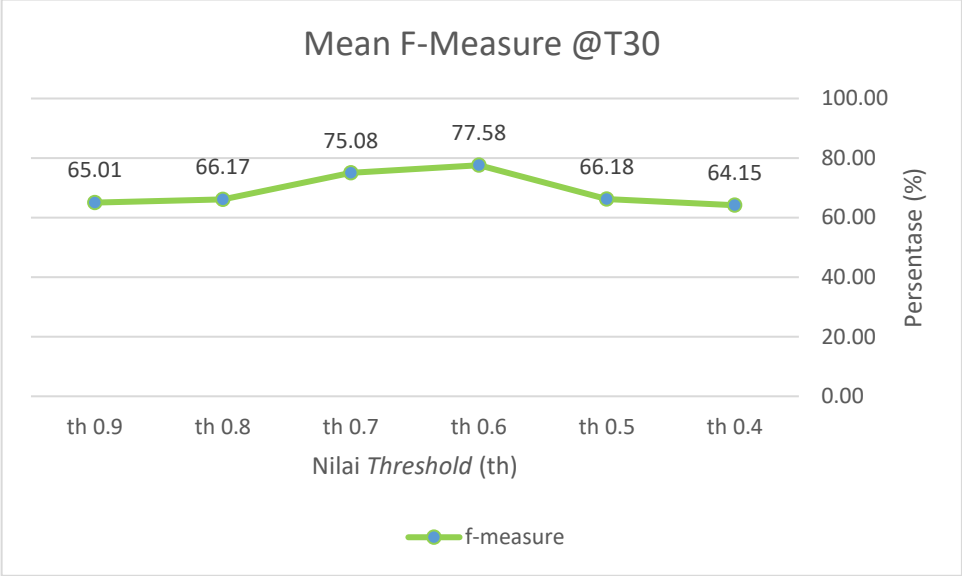
Tabel 4.8 menunjukkan nilai *mean* F-Measure pada skenario pengujian 2. Nilai F-Measure terbaik terdapat pada *threshold* 0.6 dengan nilai 77.58% sehingga dapat dikatakan rasio trade-off *Precision* dan *Recall* yang paling baik ada pada *threshold* tersebut. Tren dari nilai *mean* F-Measure dengan nilai *threshold* berbeda-beda ditunjukkan oleh Gambar 4.10.

Tabel 4.8 Nilai *Mean* F-Measure pada Skenario Uji 2

Threshold	***mean* F-Measure (%)**
th 0.9	65.01
th 0.8	66.17
th 0.7	75.08
th 0.6	**77.58**
th 0.5	66.18
th 0.4	64.15

Dengan jumlah topik T = 30, Metode usulan mempunyai nilai *Precision*, *Recall*, dan F-Measure terbaik pada *threshold* similaritas 0.6. Nilai *threshold* digunakan untuk mengambil Web Service yang relevan yang ada di repositori. Apabila menentukan *threshold* terlalu rendah maka dapat menyebabkan meningkatnya pengambilan Web Service yang tidak relevan (menurunkan

Precision), tetapi juga dapat meningkatkan banyaknya jumlah Web Service yang relevan (meningkatkan *Recall*).



Gambar 4.10 Grafik Tren Nilai F-Measure pada Skenario Uji 2

Sebaliknya, apabila menentukan *threshold* terlalu tinggi, maka dapat mengurangi banyaknya pengambilan Web Service yang tidak relevan (meningkatkan *Precision*), tetapi juga dapat mengurangi banyaknya pengambilan Web Service yang relevan (menurunkan *Recall*).

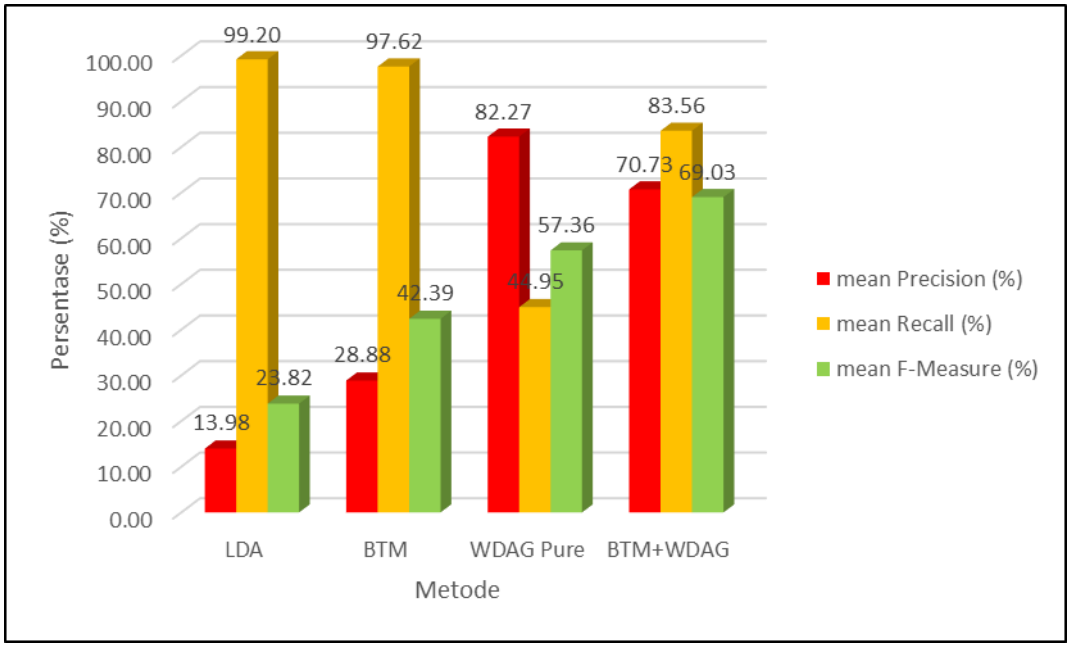
4.2.3 Analisis Hasil Skenario Pengujian 3

Dalam skenario 3, pengujian dilakukan dengan membandingkan metode usulan dengan beberapa metode. Metode yang dibandingkan yaitu LDA, BTM, dan WDAG Pure. Metode LDA dan BTM digunakan untuk menemukan web service berdasarkan kesamaan semantik saja dan Metode WDAG Pure digunakan untuk menemukan web service berdasarkan kesamaan struktur atau *behavior* saja. Hipotesa pada penelitian ini adalah dengan menggabungkan kesamaan semantik, struktur dan *behavior* dapat meningkatkan nilai *Precision*, *Recall* dan F-Measure. Tabel 4.9 menunjukkan perbandingan hasil evaluasi dari empat metode. Perbandingan nilai evaluasi secara grafis ditunjukkan oleh Gambar 4.11.

Tabel 4.9 Perbandingan Nilai Evaluasi dari 4 Metode

Metode	***mean Precision* (%)**	***mean Recall* (%)**	***mean* F-Measure (%)**
LDA	13.98	99.20	23.82
BTM	28.88	97.62	42.39
WDAG Pure	82.27	44.95	57.36
BTM+WDAG (usulan)	70.73	83.56	69.03

LDA memiliki nilai *Recall* tertinggi pertama yaitu 99.20% dikarenakan *file* WSDL atau BPEL yang relevan dengan kueri yang ada direpositori hampir semuanya dikembalikan oleh sistem. Namun LDA memiliki nilai *Precision* yang rendah yaitu 13.98% dikarenakan rasio WSDL yang tidak relevan lebih banyak dari semua *file* yang dikembalikan.



Gambar 4.11 Grafik Perbandingan Nilai Evaluasi dari 4 Metode

Kolom Web Service menunjukkan nama Web Service yang ada direpositori, kolom Sim menunjukkan nilai similaritas yang telah dihitung antara WDAG kueri dan WDAG yang ada di repositori, dan kolom Relevan menunjukkan service mana saja yang relevan dengan kueri.

Tabel 4.10 Data *Top* 50 dari kueri book_price_service.wsdl dari Metode LDA

Web Service	**Sim**	**Relevan**
2_book_price_service.wsdl.ruleml	0.973	Ya

2_book_Cheapestprice_service.wsdl.ruleml	0.972	Ya
3_book_authorprice_Novelservice.wsdl.ruleml	0.972	Ya
2_book_reviewprice_service.wsdl.ruleml	0.972	Ya
2_BookPrice.wsdl.ruleml	0.971	Ya
2_book_authorprice_service.wsdl.ruleml	0.970	Ya
2_book_authorprice_Novelservice.wsdl.ruleml	0.968	Ya
2_bookpersonOptional_price_service.wsdl.ruleml	0.960	
2_book_recommendedpriceindollar_service.wsdl.ruleml	0.959	Ya
2_bookperson_price_service.wsdl.ruleml	0.957	Ya
2_book_pricereviewbook_service.wsdl.ruleml	0.956	Ya
2_book_recommendedprice_service.wsdl.ruleml	0.952	Ya
2_bookpersoncreditcardaccount_recommendedprice_service.wsdl.ruleml	0.949	
2_book_recommendedprice_RegisteredUserservice.wsdl.ruleml	0.948	Ya
2_book_pricesizebook-type_service.wsdl.ruleml	0.948	Ya
3_book__ShoppingCartservice.wsdl.ruleml	0.948	
2_bookpersoncreditcardaccount_price_service.wsdl.ruleml	0.944	
3_book_author_service.wsdl.ruleml	0.941	
3_book_author_EncSSservice.wsdl.ruleml	0.940	
3_book_publisher_service.wsdl.ruleml	0.938	
2_bookusercreditcardaccount__service.wsdl.ruleml	0.938	
3_BookSearchService.wsdl.ruleml	0.937	
3_BookFinder.wsdl.ruleml	0.937	
3_book_person_Publisherservice.wsdl.ruleml	0.936	
2_bookpersoncreditcardaccount__BShopservice.wsdl.ruleml	0.935	
2_bookpersoncreditcardaccount__service.wsdl.ruleml	0.935	
2_bookperson__service.wsdl.ruleml	0.934	
2_bookpersoncreditaccount__Beaservice.wsdl.ruleml	0.932	
3_book_authorbook-type_service.wsdl.ruleml	0.931	
3_book_pricesizebook-type_service.wsdl.ruleml	0.926	Ya
2_book_taxedprice_service.wsdl.ruleml	0.925	Ya
2_bookpersoncreditcardaccount_taxedfreeprice_service.wsdl.ruleml	0.924	
2_bookpersoncreditaccount__service.wsdl.ruleml	0.923	
3_book_readerreview_service.wsdl.ruleml	0.923	
1_title_cdpricesoftwareStreaming_service.wsdl.ruleml	0.913	
1_title_filmpricequality_service.wsdl.ruleml	0.906	
1_TitleSaving_service.wsdl.ruleml	0.903	
1_title_media_service.wsdl.ruleml	0.896	
1_title_vhs_service.wsdl.ruleml	0.895	

1_title_filmActionComedy_service.wsdl.ruleml	0.895	
1_title_mediapricequality_service.wsdl.ruleml	0.894	
1_title_mediarecommendedpricequality_service.wsdl.ruleml	0.892	
2_book_taxedpriceprice_service.wsdl.ruleml	0.890	Ya
1_filmDiscovery_service.wsdl.ruleml	0.890	
3_book_readerreviewperson_service.wsdl.ruleml	0.889	
1_KLM-Login_service.wsdl.ruleml	0.888	
1_title_filmmaxpricequality_service.wsdl.ruleml	0.888	
1_filmHighlyRated_service.wsdl.ruleml	0.887	
3_academic-degree_funding_GermanGovservice.wsdl.ruleml	0.883	
1_title_film_service.wsdl.ruleml	0.882	

Metode BTM memiliki nilai *Recall* terbesar kedua setelah LDA dengan nilai 97.62% dan nilai *Precision* meningkat sebesar 14.9% dari LDA. Hal ini disebabkan penggalian topik pada metode BTM lebih baik dari LDA ketika diterapkan pada dokumen pendek seperti *file* WSDL atau BPEL. Tabel 4.11 menunjukkan nilai similaritas pada salah satu kueri yaitu book_price_service.wsdl pada data *top* 50 dengan menggunakan metode BTM.

Tabel 4.11 Data *Top* 50 dari kueri book_price_service.wsdl dari Metode BTM

Web service	**Sim**	**Relevan**
2_book_Cheapestprice_service.wsdl.ruleml	1	Ya
2_book_price_service.wsdl.ruleml	1	Ya
2_BookPrice.wsdl.ruleml	1	Ya
2_book_reviewprice_service.wsdl.ruleml	0.996	Ya
2_bookperson_price_service.wsdl.ruleml	0.996	Ya
2_bookpersonOptional_price_service.wsdl.ruleml	0.996	
2_book_authorprice_Novelservice.wsdl.ruleml	0.996	Ya
2_book_authorprice_service.wsdl.ruleml	0.996	Ya
3_book_authorprice_Novelservice.wsdl.ruleml	0.996	Ya
2_book_pricereviewbook_service.wsdl.ruleml	0.992	Ya
2_bookpersoncreditcardaccount_price_service.wsdl.ruleml	0.991	
2_book_recommendedprice_RegisteredUserservice.wsdl.ruleml	0.990	Ya
2_book_recommendedprice_service.wsdl.ruleml	0.990	Ya
2_bookpersoncreditcardaccount_recommendedprice_service.wsdl.ruleml	0.990	
2_bookpersoncreditcardaccount_taxedfreeprice_service.wsdl.ruleml	0.988	
2_book_taxedprice_service.wsdl.ruleml	0.987	Ya

2_book_pricesizebook-type_service.wsdl.ruleml	0.982	Ya
3_book_pricesizebook-type_service.wsdl.ruleml	0.982	Ya
3_BookFinder.wsdl.ruleml	0.980	
3_BookSearchService.wsdl.ruleml	0.980	
2_book_recommendedpriceindollar_service.wsdl.ruleml	0.975	Ya
3_book__ShoppingCartservice.wsdl.ruleml	0.968	
2_book_taxedpriceprice_service.wsdl.ruleml	0.967	Ya
3_book_publisher_service.wsdl.ruleml	0.967	
2_bookperson__service.wsdl.ruleml	0.967	
2_bookpersoncreditaccount__Beaservice.wsdl.ruleml	0.960	
2_bookpersoncreditaccount__service.wsdl.ruleml	0.960	
3_book_authorbook-type_service.wsdl.ruleml	0.958	
3_book_author_EncSSservice.wsdl.ruleml	0.958	
3_book_author_service.wsdl.ruleml	0.958	
3_book_person_Publisherservice.wsdl.ruleml	0.955	
2_bookpersoncreditcardaccount__BShopservice.wsdl.ruleml	0.952	
2_bookpersoncreditcardaccount__service.wsdl.ruleml	0.952	
2_bookusercreditcardaccount__service.wsdl.ruleml	0.946	
3_book_readerreview_service.wsdl.ruleml	0.943	
3_book_readerreviewperson_service.wsdl.ruleml	0.941	
1_TitleSaving_service.wsdl.ruleml	0.787	
1_title_cdpricesoftwareStreaming_service.wsdl.ruleml	0.761	
1_title_filmpricequality_service.wsdl.ruleml	0.725	
1_title_mediapricequality_service.wsdl.ruleml	0.721	
1_title_filmrecommendedpricequality_service.wsdl.ruleml	0.720	
1_title_filmtaxedpricequality_service.wsdl.ruleml	0.718	
1_title_film_service.wsdl.ruleml	0.718	
1_title_filmActionComedy_service.wsdl.ruleml	0.718	
1_title_filmP2P_service.wsdl.ruleml	0.718	
1_title_vhs_service.wsdl.ruleml	0.717	
1_title_mediarecommendedpricequality_service.wsdl.ruleml	0.714	
1_title_mediataxedpricequality_service.wsdl.ruleml	0.714	
1_title_media_service.wsdl.ruleml	0.712	
1_title_videomediarecommendedprice_service.wsdl.ruleml	0.708	

Metode WDAG Pure memiliki nilai *Precision* tertinggi yaitu 82.27% namun memiliki nilai *Recall* rendah yaitu 44.95%. Hal ini dikarenakan rasio *file* WSDL atau BPEL yang dikembalikan oleh sistem banyak yang relevan sehingga

nilai *Precision* tinggi, namun rasio *file* relevan yang dikembalikan oleh sistem dari total *file* relevan yang ada direpositori lebih sedikit sehingga nilai *Recall* rendah. Kelemahan dari metode WDAG Pure adalah pada saat pencocokan lable *node* menggunakan identical string matching sehingga ketika membandingkan label *node* yang menggunakan kata sinonim maka dikatakan tidak sama. Hasil similaritas pada salah satu kueri yaitu book_price_service.wsdl pada data *top* 50 dengan menggunakan metode WDAG Pure ditunjukkan oleh Tabel 4.12.

Tabel 4.12 Data *Top* 50 dari book_price_service.wsdl dari Metode WDAG Pure

Web Service	**Sim**	**Relevan**
2_book_Cheapestprice_service.wsdl.ruleml	1	Ya
2_book_price_service.wsdl.ruleml	1	Ya
2_BookPrice.wsdl.ruleml	1	Ya
2_bookperson_price_service.wsdl.ruleml	0.891	Ya
2_bookpersonOptional_price_service.wsdl.ruleml	0.891	
2_bookpersoncreditcardaccount_price_service.wsdl.ruleml	0.844	
2_car_price_service.wsdl.ruleml	0.501	
2_car2personbicycle_price_service.wsdl.ruleml	0.500	
2_carbicycle_price_service.wsdl.ruleml	0.500	
2_carcycle_price_service.wsdl.ruleml	0.500	
1_filmDiscovery_service.wsdl.ruleml	0.036	
1_filmHighlyRated_service.wsdl.ruleml	0.036	
1_TitleSaving_service.wsdl.ruleml	0.036	
1_videomediaBBC_service.wsdl.ruleml	0.036	
1_videomediaSaturn_service.wsdl.ruleml	0.036	
1_videomediaSmithLee_service.wsdl.ruleml	0.036	
1_comedyfilmactionfilm_service.wsdl.ruleml	0.034	
1_comedyfilmfantacyfilm_service.wsdl.ruleml	0.034	
1_filmvideomediaDiscoveryChannel_service.wsdl.ruleml	0.034	
1_KLM-Login_service.wsdl.ruleml	0.034	
2_bookperson__service.wsdl.ruleml	0.033	
2_bookpersoncreditaccount__Beaservice.wsdl.ruleml	0.033	
2_bookpersoncreditaccount__service.wsdl.ruleml	0.033	
2_bookpersoncreditcardaccount__BShopservice.wsdl.ruleml	0.033	
2_bookpersoncreditcardaccount__service.wsdl.ruleml	0.033	
2_bookusercreditcardaccount__service.wsdl.ruleml	0.033	
3_book__ShoppingCartservice.wsdl.ruleml	0.032	

1_linguisticexpression_videomedia_service.wsdl.ruleml	0.009	
1_title_actionfilm_service.wsdl.ruleml	0.009	
1_title_comedyfilm_BFservice.wsdl.ruleml	0.009	
1_title_comedyfilm_Megaservice.wsdl.ruleml	0.009	
1_title_comedyfilm_service.wsdl.ruleml	0.009	
1_title_film_service.wsdl.ruleml	0.009	
1_title_filmActionComedy_service.wsdl.ruleml	0.009	
1_title_filmP2P_service.wsdl.ruleml	0.009	
1_title_lowcomedyfilm_service.wsdl.ruleml	0.009	
1_title_media_service.wsdl.ruleml	0.009	
1_title_obtainablevideomedia_service.wsdl.ruleml	0.009	
1_title_vhs_service.wsdl.ruleml	0.009	
1_title_videomedia_service.wsdl.ruleml	0.009	
1_title_videomediaMM_service.wsdl.ruleml	0.009	
3_academic-degree_funding_GermanGovservice.wsdl.ruleml	0.009	
3_academic-degree_lending_GermanGovservice.wsdl.ruleml	0.009	
3_academic-degree_scholarship_GermanGovservice.wsdl.ruleml	0.009	
1_title_vhsdvd_service.wsdl.ruleml	0.007	
3_award_funding_GermanGovservice.wsdl.ruleml	0.007	
3_award_lending_GermanGovservice.wsdl.ruleml	0.007	
3_award_scholarship_GermanGovservice.wsdl.ruleml	0.007	
1_title_videomediarecommendedprice_service.wsdl.ruleml	0.006	
3_academic-degree_fundingduration_GermanGovservice.wsdl.ruleml	0.006	

Metode yang diusulkan yaitu kombinasi BTM dan WDAG (BTM+WDAG), memiliki nilai *Precision* yang lebih baik dari metode LDA dan BTM yaitu sebesar 70.73% dan memiliki nilai *Recall* yang lebih baik dari metode WDAG Pure yaitu 83.56%. Karena metode yang diusulkan mengukur kesamaan struktur antar *file* WSDL atau BPEL dengan menggunakan WDAG dan menggunakan BTM untuk mengukur kesamaan label antar *node* pada WDAG dimana metode WDAG Pure menggunakan identical string matching yang tidak bisa membedakan kata sinonim. Hasil similaritas pada salah satu kueri yaitu book_price_service.wsdl pada data *top* 50 dengan menggunakan metode usulan ditunjukkan oleh Tabel 4.13.

Tabel 4.13 Data *Top* 50 dari book_price_service.wsdl dari Metode Usulan

Web Service	**Sim**	**Relevan**
2_book_Cheapestprice_service.wsdl.ruleml	1	Ya
2_book_price_service.wsdl.ruleml	1	Ya
2_book_recommendedprice_RegisteredUserservice.wsdl.ruleml	1	Ya
2_book_recommendedprice_service.wsdl.ruleml	1	Ya
2_book_recommendedpriceindollar_service.wsdl.ruleml	1	Ya
2_BookPrice.wsdl.ruleml	1	Ya
2_book_taxedprice_service.wsdl.ruleml	0.927	Ya
2_book_reviewprice_service.wsdl.ruleml	0.891	Ya
2_bookperson_price_service.wsdl.ruleml	0.891	Ya
2_bookpersonOptional_price_service.wsdl.ruleml	0.891	
2_bookpersoncreditcardaccount_price_service.wsdl.ruleml	0.844	
2_bookpersoncreditcardaccount_recommendedprice_service.wsdl.ruleml	0.844	
2_bookpersoncreditcardaccount_taxedfreeprice_service.wsdl.ruleml	0.844	
2_book_pricereviewbook_service.wsdl.ruleml	0.834	Ya
2_book_authorprice_Novelservice.wsdl.ruleml	0.828	Ya
2_book_authorprice_service.wsdl.ruleml	0.828	Ya
3_book_authorprice_Novelservice.wsdl.ruleml	0.828	Ya
2_book_taxedpriceprice_service.wsdl.ruleml	0.823	Ya
2_book_pricesizebook-type_service.wsdl.ruleml	0.781	Ya
3_book_pricesizebook-type_service.wsdl.ruleml	0.781	Ya
2_carbicycle_price_service.wsdl.ruleml	0.530	
2_carbicycle_recommendedprice_service.wsdl.ruleml	0.530	
2_carcycle_price_service.wsdl.ruleml	0.530	
2_car2personbicycle_price_service.wsdl.ruleml	0.522	
2_car_price_service.wsdl.ruleml	0.519	
2_car_recommendedprice_service.wsdl.ruleml	0.519	
2_car_recommendedpriceindollar_service.wsdl.ruleml	0.519	
2_car_recommendedpriceineuro_service.wsdl.ruleml	0.519	
3_book__ShoppingCartservice.wsdl.ruleml	0.506	
2_butter_maxprice_service.wsdl.ruleml	0.500	
2_butter_recommendedprice_service.wsdl.ruleml	0.500	
2_carbicycle_taxedprice_service.wsdl.ruleml	0.457	
2_butter_taxedprice_service.wsdl.ruleml	0.428	
2_car_pricequality_service.wsdl.ruleml	0.410	
1_title_cdpricesoftwareStreaming_service.wsdl.ruleml	0.398	
2_bookperson__service.wsdl.ruleml	0.396	

2_car_priceauto_service.wsdl.ruleml	0.395	
2_car_yearprice_service.wsdl.ruleml	0.394	
2_car_pricereport_service.wsdl.ruleml	0.394	
3_beverage_maxpricequantity_service.wsdl.ruleml	0.390	
3_beverage_pricequantity_Aldiservice.wsdl.ruleml	0.390	
3_beverage_taxfreepricequantity_service.wsdl.ruleml	0.390	
3_beverage_maxpricephysical-quantity_service.wsdl.ruleml	0.384	
3_beverage_pricephysical-quantity_Aldiservice.wsdl.ruleml	0.384	
3_beverage_taxfreepricephysical-quantity_service.wsdl.ruleml	0.384	
2_breadorbiscuit_recPricetaxedpriceineuro_service.wsdl.ruleml	0.377	
2_bookpersoncreditaccount__Beaservice.wsdl.ruleml	0.350	
2_bookpersoncreditaccount__service.wsdl.ruleml	0.350	
2_bookpersoncreditcardaccount__BShopservice.wsdl.ruleml	0.350	
2_bookpersoncreditcardaccount__service.wsdl.ruleml	0.350	

4.2.4 Analisis Hasil Terjadinya Kesalahan *Retrieve*

Metode yang diusulkan, yaitu kombinasi BTM dan WDAG Similarity, menghasilkan nilai *Precision* dan *Recall* yang tidak 100% dikarenakan hasil keluaran metode yang diusulkan masih terjadi ketidaksesuaian dalam pengembalian Web Service yang mirip. Hal ini disebabkan oleh terjadinya kesalahan penggalian topik dari metode BTM, dimana node label yang seharusnya mirip dikatakan tidak mirip karena hasil distribusi probabilitas topik yang jauh berbeda. Kesalahan dalam mencari node label yang mirip mempengaruhi performa dari metode WDAG Similarity sehingga dapat menurunkan nilai similaritas antara dua WDAG yang seharusnya memiliki nilai similaritas yang tinggi. Sehingga masih diperlukan penelitian selanjutnya untuk memperbaiki pengukuran kesamaan kata secara semantik.

[halaman sengaja dikosongkan]

BAB 5
PENUTUP

Bab ini berisi kesimpulan dari penelitian yang telah dilakukan beserta saran untuk pengembangan lebih lanjut.

5.1 Kesimpulan

Berdasarkan hasil uji coba yang dilakukan pada penelitian ini dapat disimpulkan beberapa hal.

- Metode penemuan web service yang diusulkan, yaitu metode kombinasi BTM dan WDAG Similarity, memiliki rata-rata *Precision* sebesar 70.73%, rata-rata *Recall* sebesar 83.56%, dan rata-rata F-Measure sebesar 69.03% dimana jumlah Topik yang digali sebesar 30 Topik. Banyaknya jumlah topik dapat mempengaruhi nilai *Precision*, *Recall*, dan F-Measure. Apabila terlalu sedikit menentukan jumlah topik maka dapat menyebabkan label *node* yang seharusnya berbeda topik justru memiliki topik yang sama. Apabila terlalu banyak menentukan jumlah topik maka dapat menyebabkan label *node* yang seharusnya memiliki topik yang sama justru memiliki topik yang berbeda. Sehingga diperlukan metode untuk menentukan jumlah topik yang tepat sesuai dengan dataset yang digunakan.
- Dengan jumlah topik T = 30, Metode usulan mempunyai nilai *Precision*, *Recall*, dan F-Measure terbaik pada *threshold* similaritas 0.6. Nilai *threshold* digunakan untuk mengambil Web Service yang relevan yang ada di repositori. Apabila menentukan *threshold* terlalu rendah maka dapat menyebabkan meningkatnya pengambilan Web Service yang tidak relevan (menurunkan *Precision*), tetapi juga dapat meningkatkan banyaknya jumlah Web Service yang relevan (meningkatkan *Recall*). Sebaliknya, apabila menentukan *threshold* terlalu tinggi, maka dapat mengurangi banyaknya pengambilan Web Service yang tidak relevan (meningkatkan *Precision*), tetapi juga dapat mengurangi banyaknya pengambilan Web Service yang relevan (menurunkan *Recall*). Sehingga diperlukan metode untuk penentuan *threshold* yang dinamis berdasarkan nilai *Precision* dan *Recall*.

- Metode yang diusulkan memiliki rasio trade-off antara *Precision* dan *Recall* terbaik (F-Measure) daripada 3 metode lain yaitu LDA, BTM dan WDAG Pure sebesar 69.03%. Metode usulan unggul dari segi *Precision* dari metode LDA dan BTM dengan nilai sebesar 70.73% dan unggul dari segi nilai *Recall* dari Metode WDAG Pure dengan nilai sebesar 83.56%.
- Pada metode yang diusulkan masih terjadi ketidaksesuaian pengembalian Web Service yang mirip dikarenakan terjadinya kesalahan penggalian topik dari metode BTM, dimana node label yang seharusnya mirip dikatakan tidak mirip karena hasil distribusi probabilitas topik yang jauh berbeda.

5.2 Saran

Berdasarkan hasil pengujian, untuk penelitian selanjutnya, perlu dilakukan penentuan *threshold* yang dinamis yang dapat berubah-ubah dengan membandingkan banyaknya ratio data yang relevan dengan data yang tidak relevan yang dapat dikembalikan sistem, sehingga nilai *Precision* dan *Recall* yang dihasilkan optimal. Penemuan web service dengan menggunakan metode topik modelling juga perlu menentukan jumlah topik yang tepat dalam memodelkan distribusi probabilitas topik supaya nilai *Precision* dan *Recall* optimal.

Secara keseluruhan, metode yang diusulkan sudah menghasilkan hasil yang lebih baik dari metode lain yang dibandingkan. Namun, masih dapat ditingkatkan dengan melakukan penelitian lebih lanjut pada pengukuran kesamaan kata secara semantik. Pada metode yang diusulkan masih terjadi ketidaksesuain dalam pengembalian Web Service dikarenakan terjadinya kesalahan penggalian topik dari metode BTM, dimana node label yang seharusnya mirip dikatakan tidak mirip karena hasil distribusi probabilitas topik yang jauh berbeda.

DAFTAR PUSTAKA

Aznag, Mustapha, Mohamed Quafafou, and Zahi Jarir. 2013. "Correlated Topic Model for Web Services Ranking." *International Journal of Advanced Computer Science and Applications* 4(6):283–91.

Blei, David M., Andrew Y. Ng, and Michael I. Jordan. 2003. "Latent Dirichlet Allocation." *Journal of Machine Learning Research* 3(4-5):993–1022.

Bravo, Maricela. 2014. "SIMILARITY MEASURES FOR WEB SERVICE." *International Journal on Web Service Computing* 5(1):1–16.

Grigori, Daniela, Juan Carlos Corrales, Mokrane Bouzeghoub, and Ahmed Gater. 2010. "Ranking BPEL Processes for Service Discovery." Pp. 178–92 in *IEEE Transactions on Services Computing*, vol. 3. IEEE.

Gunay, Akin and Pinar Yolum. 2007. "Structural and Semantic Similarity Metrics for Web Service Matchmaking." Pp. 129–38 in *Lecture Notes in Computer Science (LNCS)*.

Halevy, Alon, Ema Nemes, Xin Dong, Jayant Madhavan, and Jun Zhang. 2004. "Similarity Search for Web Services." Pp. 372–83 in *Proceedings of the 30th VLDB Conference*. Toronto.

Hou, Jun, Jinglan Zhang, Richi Nayak, and Aishwarya Bose. 2010. "Semantics-Based Web Service Discovery Using." Pp. 336–46 in *International Workshop of the Initiative for the Evaluation of XML Retrieval*.

Jin, Jing. 2006. "Similarity of Weighted Directed Acyclic Graph." University of New Brunswick.

Lei, Yu and Philip S. Yu. 2016. "Service Topic Model with Probability Distance." Pp. 202–7 in *9th International Conference on Utility and Cloud Computing*. IEEE/ACM.

Li, Chune, Richong Zhang, Jinpeng Huai, Xiaohui Guo, and Hailong Sun. 2013. "A Probabilistic Approach for Web Service Discovery." Pp. 49–56 in *IEEE*

International Conference on Services Computing. IEEE.

Maheswari, J. Uma. 2014. “Comparison of Web Service Similarity- Assessment Methods.” *International Journal of Computer Applicaitons* 98(22):18–23.

Paul, Arnab, Sudipta Roy, and Monowar Hussain. 2016. “Web Service Discovery Based on IR Models : A Review.” in *International Conference on Communication and Electronics Systems (ICCES)*. IEEE.

Platzer, Christian and Schahram Dustdar. 2005. “A Vector Space Search Engine for Web Services.” Pp. 62–71 in *Proceedings of the Third European Conference on Web Services*.

Qin, Zuoyan, Peng Li, Qing Zhu, and Chao Tian. 2010. “SWEE : Approximately Searching Web Service with Keywords Effectively and Efficiently.” Pp. 569–74 in *2nd International Conference on Advanced Computer Control*.

Stavropoulos, Thanos G., Stelios Andreadis, Nick Bassiliades, Dimitris Vrakas, and Ioannis Vlahavas. 2016. “The Tomaco Hybrid Matching Framework for SAWSDL Semantic Web Services.” *IEEE Transactions on Services Computing* 9(6):954–67.

Steyvers, Mark and Tom Griffiths. 2007. “Probabilistic Topic Models.” *Handbook of latent semantic analysis* 427(7):424–40.

Wan, Xiaomin, Xiaoguang Mao, and Ziying Dai. 2012. “Mining and Checking Web Service Behavior.” Pp. 1004–9 in *International Conference on Industrial Informatics (INDIN)*.

Wang, Jian, Panpan Gao, Yutao Ma, and Keqing He. 2015. “Common Topic Group Mining for Web Service Discovery.” Pp. 92–107 in *Asia-Pacific Services Computing Conference*.

Wu, Chen, Vidyasagar Potdar, and Elizabeth Chang. 2008. “Latent Semantic Analysis – The Dynamics of Semantics Web Services Discovery.” *Lecture Notes in Computer Science (LNCS) on Advances in Web Semantics* 4891:346–73.

Yan, Xiaohui, Jiafeng Guo, Yanyan Lan, and Xueqi Cheng. 2013. “A Biterm Topic Model for Short Texts.” Pp. 1445–55 in *Proceedings of the 22nd international conference on World Wide Web*. Rio De Janeiro.

Zhang, Neng, Jian Wang, Keqing He, and Zheng Li. 2016. “An Approach of Service Discovery Based on Service Goal Clustering.” Pp. 114–21 in *International Conference on Services Computing*. IEEE.

Zinnikus, Ingo, Hans-Jorg Rupp, and Klaus Fischer. 2010. “Detecting Similarities between Web Service Interfaces: The WSDL Analyzer.” in *Interoperability for Enterprise Software and Applications: Proceedings of the Workshops and the Doctorial Symposium of the Second IFAC/IFIP I-ESA International Conference*.

[halaman sengaja dikosongkan]

LAMPIRAN

Tabel 7.1 Daftar Web Service dengan Domain *Car Service*

No.	Nama Web Service
1	2_car2personbicycle_price_service.wsdl
2	2_car_price_service.wsdl
3	2_car_priceauto_service.wsdl
4	2_car_pricecolor_service.wsdl
5	2_car_pricequality_service.wsdl
6	2_car_pricereport_service.wsdl
7	2_car_recommendedprice_service.wsdl
8	2_car_recommendedpriceindollar_service.wsdl
9	2_car_recommendedpriceineuro_service.wsdl
10	2_car_report_service.wsdl
11	2_car_taxedpriceprice_service.wsdl
12	2_car_taxedpricereport_service.wsdl
13	2_car_yearprice_service.wsdl
14	2_carbicycle_price_service.wsdl
15	2_carbicycle_recommendedprice_service.wsdl
16	2_carbicycle_taxedprice_service.wsdl
17	2_carcycle_price_service.wsdl
18	3_auto_technology_service.wsdl

Tabel 7.2 Daftar Web Service dengan Domain *Book Service*

No.	Nama Web Service
1	2_book_authorprice_Novelservice.wsdl
2	2_book_authorprice_service.wsdl
3	2_book_Cheapestprice_service.wsdl
4	2_book_price_service.wsdl
5	2_book_pricereviewbook_service.wsdl
6	2_book_pricesizebook-type_service.wsdl
7	2_book_recommendedprice_RegisteredUserservice.wsdl
8	2_book_recommendedprice_service.wsdl
9	2_book_recommendedpriceindollar_service.wsdl
10	2_book_reviewprice_service.wsdl
11	2_book_taxedprice_service.wsdl
12	2_book_taxedpriceprice_service.wsdl

13	2_bookperson__service.wsdl
14	2_bookperson_price_service.wsdl
15	2_bookpersoncreditaccount__Beaservice.wsdl
16	2_bookpersoncreditaccount__service.wsdl
17	2_bookpersoncreditcardaccount__BShopservice.wsdl
18	2_bookpersoncreditcardaccount__service.wsdl
19	2_bookpersoncreditcardaccount_price_service.wsdl
20	2_bookpersoncreditcardaccount_recommendedprice_service.wsdl
21	2_bookpersoncreditcardaccount_taxedfreeprice_service.wsdl
22	2_bookpersonOptional_price_service.wsdl
23	2_BookPrice.wsdl
24	2_bookusercreditcardaccount__service.wsdl
25	3_book__ShoppingCartservice.wsdl
26	3_book_author_EncSSservice.wsdl
27	3_book_author_service.wsdl
28	3_book_authorbook-type_service.wsdl
29	3_book_authorprice_Novelservice.wsdl
30	3_book_authortext_service.wsdl
31	3_book_person_Publisherservice.wsdl
32	3_book_pricesizebook-type_service.wsdl
33	3_book_publisher_service.wsdl
34	3_book_readerreview_service.wsdl
35	3_book_readerreviewperson_service.wsdl
36	3_BookFinder.wsdl
37	3_BookSearchService.wsdl
38	Book01.bpel
39	Book02.bpel
40	Book03.bpel
41	Book04.bpel
42	Book05.bpel
43	Book06.bpel
44	Book07.bpel

Tabel 7.3 Daftar Web Service dengan Domain *Film Service*

No.	Nama Web Service
1	1_comedyfilmactionfilm_service.wsdl
2	1_comedyfilmfantacyfilm_service.wsdl
3	1_filmDiscovery_service.wsdl

4	1_filmHighlyRated_service.wsdl
5	1_filmvideomediaDiscoveryChannel_service.wsdl
6	1_KLM-Login_service.wsdl
7	1_linguisticexpression_videomedia_service.wsdl
8	1_title_actionfilm_service.wsdl
9	1_title_actionfilmmaxpricequality_service.wsdl
10	1_title_actionfilmpricequality_service.wsdl
11	1_title_actionfilmrecommendedpricequality_service.wsdl
12	1_title_actionfilmtaxedpricequality_service.wsdl
13	1_title_actionfilmtaxfreepricequality_service.wsdl
14	1_title_cdpricesoftwareStreaming_service.wsdl
15	1_title_comedyfilm_BFservice.wsdl
16	1_title_comedyfilm_Megaservice.wsdl
17	1_title_comedyfilm_service.wsdl
18	1_title_comedyfilmmaxpricequality_service.wsdl
19	1_title_comedyfilmpricequality_service.wsdl
20	1_title_comedyfilmrecommendedpricequality_service.wsdl
21	1_title_comedyfilmtaxedpricequality_service.wsdl
22	1_title_comedyfilmtaxfreepricequality_service.wsdl
23	1_title_film_service.wsdl
24	1_title_filmActionComedy_service.wsdl
25	1_title_filmmaxpricequality_service.wsdl
26	1_title_filmP2P_service.wsdl
27	1_title_filmpricequality_service.wsdl
28	1_title_filmrecommendedpricequality_service.wsdl
29	1_title_filmtaxedpricequality_service.wsdl
30	1_title_filmtaxfreepricequality_service.wsdl
31	1_title_highcomedyfilmreport_service.wsdl
32	1_title_lowcomedyfilm_service.wsdl
33	1_title_media_service.wsdl
34	1_title_mediamaxpricequality_service.wsdl
35	1_title_mediapricequality_service.wsdl
36	1_title_mediarecommendedpricequality_service.wsdl
37	1_title_mediataxedpricequality_service.wsdl
38	1_title_mediataxfreepricequality_service.wsdl
39	1_title_obtainablevideomedia_service.wsdl
40	1_title_sciencefictionfilmmaxpricequality_service.wsdl
41	1_title_sciencefictionfilmpricequality_service.wsdl

42	1_title_sciencefictionfilmrecommendedpricequality_service.wsdl
43	1_title_sciencefictionfilmtaxedpricequality_service.wsdl
44	1_title_sciencefictionfilmtaxfreepricequality_service.wsdl
45	1_title_vhs_service.wsdl
46	1_title_vhsdvd_service.wsdl
47	1_title_videomedia_service.wsdl
48	1_title_videomediamaxpricequality_service.wsdl
49	1_title_videomediaMM_service.wsdl
50	1_title_videomediapricequality_service.wsdl
51	1_title_videomediarecommendedprice_service.wsdl
52	1_title_videomediarecommendedpricequality_service.wsdl
53	1_title_videomediataxedpricequality_service.wsdl
54	1_title_videomediataxfreepricequality_service.wsdl
55	1_TitleSaving_service.wsdl
56	1_videomediaBBC_service.wsdl
57	1_videomediaSaturn_service.wsdl
58	1_videomediaSmithLee_service.wsdl

Tabel 7.4 Daftar Web Service dengan Domain *Food* Service

No.	Nama Web Service
1	2_breadorbiscuit_recPricetaxedpriceineuro_service.wsdl
2	2_butter_maxprice_service.wsdl
3	2_butter_recommendedprice_service.wsdl
4	2_butter_taxedprice_service.wsdl
5	3_beverage_maxpricephysical-quantity_service.wsdl
6	3_beverage_maxpricequantity_service.wsdl
7	3_beverage_pricephysical-quantity_Aldiservice.wsdl
8	3_beverage_pricequantity_Aldiservice.wsdl
9	3_beverage_taxedpricephysical-quantity_service.wsdl
10	3_beverage_taxedpricequantity_service.wsdl
11	3_beverage_taxfreepricephysical-quantity_service.wsdl
12	3_beverage_taxfreepricequantity_service.wsdl

Tabel 7.5 Daftar Web Service dengan Domain *Academic Service*

No.	Nama Web Service
1	3_academic-degree_funding_GermanGovservice.wsdl
2	3_academic-degree_fundingduration_GermanGovservice.wsdl

3	3_academic-degree_lending_GermanGovservice.wsdl
4	3_academic-degree_lendingduration_GermanGovservice.wsdl
5	3_academic-degree_scholarship_GermanGovservice.wsdl
6	3_academic-degree_scholarshipduration_GermanGovservice.wsdl
7	3_academic-degreegovernment_funding_service.wsdl
8	3_academic-degreegovernment_lending_service.wsdl
9	3_academic-degreegovernment_scholarship_service.wsdl
10	3_academic-degreegovernment_unilateralgiving_service.wsdl
11	3_academic-degreegovernmentorganization_funding_service.wsdl
12	3_academic-degreegovernmentorganization_lending_service.wsdl
13	3_academic-degreegovernmentorganization_unilateralgiving_service.wsdl
14	3_academic-item-number_book_service.wsdl
15	3_academic-item-number_bookauthor_service.wsdl
16	3_academic-item-number_publication_service.wsdl
17	3_academic-item-number_publicationauthor_service.wsdl
18	3_academic_address_service.wsdl
19	3_academic_postal-address_service.wsdl
20	3_AcademicBookNumberOrISBNSearch.wsdl
21	3_AcademicBookNumberSearch.wsdl

Tabel 7.6 Daftar Web Service dengan Domain *Government Service*

No.	Nama Web Service
1	3_award_funding_GermanGovservice.wsdl
2	3_award_fundingduration_GermanGovservice.wsdl
3	3_award_lending_GermanGovservice.wsdl
4	3_award_lendingduration_GermanGovservice.wsdl
5	3_award_scholarship_GermanGovservice.wsdl
6	3_award_scholarshipduration_GermanGovservice.wsdl
7	3_award_scholarshipduration_SwissGovservice.wsdl
8	3_awardgovernment_funding_service.wsdl

Tabel 7.7 Daftar Web Service dengan Domain *Flight Service*

No.	Nama Web Service
1	FlightReservation01.bpel
2	FlightReservation02.bpel
3	FlightReservation03.bpel
4	FlightReservation04.bpel

5	FlightReservation05.bpel
6	FlightReservation06.bpel

Tabel 7.8 Daftar Web Service dengan Domain *Hotel Service*

No.	Nama Web Service
1	HotelReservation01.bpel
2	HotelReservation02.bpel
3	HotelReservation03.bpel
4	HotelReservation04.bpel
5	HotelReservation05.bpel
6	Payment01

BIODATA PENULIS

Andreyan Rizky Baskara lahir di Banjarmasin pada tanggal 3 Juli 1993, Anak kedua dari dua bersaudara. Pendidikan Strata 1 di Institut Teknologi Sepuluh Nopember Fakultas Teknologi Informasi Jurusan Teknik Informatika dengan bidang keahlian Rekayasa Perangkat Lunak dan lulus tahun 2014.

Kemudian pada tahun 2015, penulis diterima di Magister Teknik Informatika Fakultas Teknologi Informasi ITS melalui jalur mandiri dan berhasil menyelesaikan studi pada tahun 2017 dengan bidang keahlian RPL (Rekayasa Perangkat Lunak). Penulis memiliki ketertarikan dalam bidang pembuatan perangkat lunak, dan pemodelan. Penulis dapat dihubungi melalui email: andreyanrb@gmail.com

[halaman sengaja dikosongkan]